Servicio web de Carpooling

Autores:

Guillermo Conesa Esteban
Juan Ignacio Martínez Amat
Ignacio Serrano Gómez

SISTEMAS INFORMÁTICOS. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Julio 2012

Directores:

José Ignacio Gómez
Christian Tenllado

Guillermo Conesa Esteban ,Juan Ignacio Martínez Amat y Ignacio Serrano Gómez autorizamos a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado.

Fdo. Guillermo Conesa Fdo. Juan Ignacio Martínez Fdo. Ignacio Serrano

Palabras clave:

- Coche compartido
- Aplicación web
- Trayecto
- Ecológico
- Google Maps

Keywords:

- Carpooling
- Web application
- Journey
- Ecological
- Google Maps

Índice general

Capítulo 1

Resumen

Español

El Carpooling es la práctica de compartir el uso de un automóvil por dos o más personas, generalmente para viajar juntos durante las horas pico hacia el trabajo o un centro educativo. Esta práctica también es utilizada para los viajes de media y larga distancia, ya sea por cuestiones laborales o de placer.

Abordamos este proyecto con el fin de conseguir una serie de objetivos como son la disminución de la emisión de CO2 y la contaminación, desplazamientos más eficientes para reducir el tráfico o el ahorro en combustible, para así alcanzar una serie de ventajas y beneficios tanto para el usuario como para la sociedad.

Para todo esto hemos utilizado y perfeccionado conocimientos adquiridos en la carrera como son la gestión de bases de datos, metodologías de programación así como algoritmos concretos estudiados en informática gráfica, ingeniería del software, patrones de diseño, etc... así como otras tecnologías en las que no se profundiza en la carrera como puede ser la programación web y diferentes tecnologías, como son Google Maps, php, AJAX,etc...

Inglés

Carpooling is the sharing of the use of a car by two or more people traveling together usually during peak hours; normally when it is time to go to work or to school. This practice is also used for medium and long travel distances, whether for business or pleasure.

This project was approached in order to achieve a number of objectives such as: the reduction of CO2 emissions and pollution and a more efficient travel to reduce traffic jams and fuel savings. The purpose is to benefit the users of Carpooling and the society in general.

In order to be able to develop this project, we have used and improve the knowledge acquired during the career. We have applied what we have learnt regarding database management, programming methodologies, specific algorithms studied in computer graphics, software engineering, design patterns, and some others. In addition, we also used some technologies that were not discussed during the career such as web programming, Google Maps, PHP, AJAX...

Capítulo 2

Introducción

En este capítulo se describirá de manera general el problema que se pretende resolver. Asimismo se intentará arrojar luz sobre algunos conceptos que se repetirán en capítulos posteriores, facilitando así la comprensión del documento.

2.1. Motivación

El transporte privado en las grandes ciudades se ha convertido en un gran problema, ya que la cantidad de vehículos en circulación tiende a aumentar. El aumento de infraestructuras muchas veces no es una solución, ya no solo por lo económico sino por el impacto que tiene esto en el medioambiente. Es necesario un cambio cultural, una nueva forma de organizarse para reducir el tráfico y la contaminación. Existen otras formas de lograr esto que no sean aumentar la capacidad de las carreteras. Consiste en lograr un sistema de transporte sostenible, promoviendo medios de transporte con bajo impacto medioambiental.

Nosotros pensamos que aprovechar las plazas libres existentes en los vehículos es la mejor solución posible para este problema. Es así como surge nuestro proyecto, que consiste en implementar una aplicación web que cumpla la función de facilitar el contacto entre personas que estén interesadas en compartir coche, ya sea para ir a trabajar, a la universidad, o viajes de larga distancia. El proyecto se enmarca no solo en la idea de una mejor gestión de la

movilidad en cuanto al vehículo privado se refiere, sino entendiéndolo como una primera parte de una acción generadora de los necesarios cambios sociales que debemos afrontar en el insostenible modelo de movilidad actual.
En concreto, el carpooling se basa en:

- Potenciar un uso más racional del coche
- Menor contaminación atmosférica y de ruido
- Mejor uso del espacio público, con disminución del grado de congestión en carreteras y vías urbanas.
- Ahorro de combustible y compartir gastos

Generalmente, este sistema se utiliza en los viajes de ida y vuelta hacia los centros de trabajo o centros educativos, pero también es aplicable a viajes de largo recorrido. La máxima eficiencia del carpooling se produce cuando se cumplen las siguientes tres condiciones:

- Proximidad del origen y destino del viaje de los ocupantes del coche (o en su defecto, que el origen y destino de los ocupantes se encuentren a lo largo de un mismo recorrido).
- Coincidencia de horarios a la ida y a la vuelta.
- Viajes de tipo recurrente, por motivos de trabajo o estudios.

2.2. Origen del carpooling

La idea del carpooling no es algo nuevo, ya que se viene realizando en EEUU desde la segunda guerra mundial, como una estrategia de racionamiento del petróleo. Durante la cual, se animaba a los americanos a ser patriotas e intentar ser austeros en su consumo, compartiendo coche con sus vecinos con campañas de publicidad. Aunque fue con la crisis del petróleo de 1973 que permitió verdaderamente el desarrollo del carpooling, tras la decisión de la Organización de Países Exportadores de Petróleo de no exportar más petróleo a los países

que habían apoyado a Israel durante la guerra del Yom Kippur, que enfrentaba a Israel con Siria y Egipto. Esta medida incluía a EEUU, y a sus aliados de Europa Occidental. Fue en aquella época cuando mayor auge alcanzo el carpooling con la construcción de los primeros carriles para vehículos de alta ocupación (carriles VAO), llegando a ser en 1980 la forma de ir a trabajar para un 20 % de los americanos. Sin embargo, desde entonces, al mismo ritmo que los precios de los coches y la gasolina disminuían, también lo hacia la práctica del carpooling, llegando a ser en 2009 la forma de ir a trabajar del 10 % de los americanos.

Mientras tanto, en Europa hasta los años noventa no se empezó a desarrollarse los viajes en coches compartidos. Especialmente en Holanda, que lanzó una campaña de información a nivel nacional, o en Bélgica, la cual ofrece una base de datos nacional a las empresas que quieren fomentar este modo de transporte.

La situación en España es bien distinta. Mas del 63 % de los españoles utilizan a diario el coche para ir a trabajar, con una tasa de ocupación de 1,2 personas por coche. Este escaso aprovechamiento del transporte privado hace que aquellos que se desplacen en coche ocupen 90 veces más espacio que aquellos otros que hagan este mismo desplazamiento en metro, y 20 veces más que quienes lo hagan en autobús.

El tráfico es el responsable del 75 % de las emisiones totales de C02 y del 80 % del ruido urbano. En las áreas metropolitanas, el consumo energético medio y las emisiones de C02 por viajero en distancias menores de 10 kilómetros son entre 2 y 3 veces superiores cuando el trayecto se hace en automóvil, que cuando se realiza en autobús o metro.

2.3. Distinción entre Carpooling y Carsharing

Cuando se habla de coche compartido existe una pequeña confusión entre dos conceptos distintos, el carsharing y el carpooling; ya que se traducen ambos conceptos por "compartir coche", mientras que se trata de cosas muy distintas. Carsharing consiste en la multipropiedad de un coche o uso alternativo del mismo, mientras que carpooling consiste en compartir trayectos de coche. El carsharing tiene obviamente su interés desde el punto de vista económico para sus co-propietarios o co-usuarios. Supongamos dos personas que tienen unas necesidades de desplazamiento complementarias (es decir, no solapadas en el tiempo), por ejemplo uno necesita el coche los días laborables y el otro los fines de semana y festivos. Pues bien, ¿para qué comprar dos coches si con uno basta? Por tanto, cuando nos refiramos a coche compartido a lo largo de la memoria, nos estaremos refiriendo a carpooling.

2.4. Ventajas de un sistema de Carpooling

Las ventajas que se pueden sacar del uso del Carpooling son:

2.4.1. Ventajas para la empresa

Para una empresa, las ventajas que puede sacar son:

- Permite el ahorro de costes a través de la supresión o de la reducción de los gastos de las plazas de parking.
- Contribuye a los objetivos propios de la empresa de desarrollo sostenible o medioambiental.
- Demuestra su implicación en la responsabilidad social.
- Mejora la imagen de la empresa
- Crea y mejora de las redes informales internas

- Mejora su productividad, ya que sus trabajadores llegarían en menos tiempo debido a la reducción en el trafico, y con mejor predisposición a trabajar, lo que afecta directamente a los intereses de la empresa.

2.4.2. Ventajas para la sociedad

Para la sociedad, las ventajas identificadas del viaje en coche compartido son:

- Reducción de la congestión urbana.
- Reducción del nivel de contaminación.
- Reducción del consumo global de gasolina.

2.4.3. Ventajas para los empleados

Las ventajas para los empleados son las detalladas a continuación:

- Ahorro significativo de los gastos, dado que el coste de la gasolina, parking, y del funcionamiento del vehículo se comparten.
- Ahorro de tiempo si la empresa proporciona la disponibilidad de plazas de parking prioritarias.

2.5. Servicios parecidos de Carpooling en España:

A continuación ponemos servicios similares al nuestro:

- El único antecedente anterior a la existencia de Internet corresponde a la iniciativa realizada por el Centro de Viaje Compartido (CVC) del departamento de Urbanística y Ordenación del Territorio de la E.T.S. de Arquitectura de la Universidad Politécnica de Madrid en el año 1997. Dicha iniciativa buscaba fomentar la utilización de los carriles de alta ocupación construidos en la A-6 en 1995, mediante la práctica de coche compartido. Se ofrecía un sistema de subscripción, a mano, mediante formularios en

los que los interesados se apuntaban a la iniciativa. Después el CVC les entregaba una lista de contactos compatibles con su trayecto y, ellos mismos, llamándose por teléfono debían concretar el viaje compartido. Como se puede observar, el sistema es un precursor arcaico de lo que se realiza (y pretendemos realizar) actualmente de forma online.

- www.compartir.org Es una red o directorio de municipios, empresas y organismos que fomentan el servicio de compartir coche. Está presente en más de 70 países.
- www.carpoolworld.com Es una empresa cuya página web sirve de punto de encuentro de las personas interesadas en compartir coche
- www.carpoolconnect.com Es otra página web con características similares a la anterior
- Zimride Carpool es una aplicación para la red social Facebook que se encarga de organizar todos aquellos usuarios de dicha red interesados en el carpooling
- www.viajamosjuntos.com Creada en 2004
- www.comuto.es Creada en 2011, incluye la posibilidad de registrarse de forma vinculada a Facebook. Sirve para viajes de larga distancia (entre ciudades), para una fecha relativamente lejana, dando tiempo así a ser respondido el anuncio. La página esta también asociada a eventos y festivales a los que se sugiere y anima a los participantes a asistir de esta forma.
- www.ofimovi-comparte.es Es una iniciativa pública de la oficina de movilidad de Burgos bajo el lema: "te muevo".
- www.carpooling.es Es una página web que se dedica a poner en contacto conductores con pasajeros por toda España. Si vas a cualquier parte de Europa, también puedes seguir utilizando sus servicios. Es la Red social de movilidad número uno en Europa.

Colaboran con el programa del Gobierno de España para promover el Carpooling www.compartescoche.es.

- www.amovens.com Iniciativa que se caracteriza porque ofrece soluciones para empresas, universidades, administraciones y para eventos (deportivos, festivales de música,...).
- www.menoshumos.es Es una acción del Plan de Acción del Gobierno de Aragón frente al Cambio Climático incluida también en el Plan Integral de Seguridad Vial de Aragón.
- www.aplicacionesua.cpd.ua.es/autocolega Una Iniciativa de la Universidad de Alicante para fomentar el uso del "coche compartido" para las rutas universitarias.

Capítulo 3

Objetivos y requisitos

3.1. Objetivos generales

El objetivo principal del proyecto es desarrollar una aplicación web que permita poner en contacto a un número de personas para la racionalización en el uso del vehículo y así fomentar la idea del compartimiento de coches.

Por estos objetivos descritos se derivan otros como realizar una aplicación que resulte útil para la sociedad y así poder mostrar como las nuevas tecnologías pueden ayudar en los problemas que día a día nos encontramos en la sociedad (contaminación, falta de recursos...).

3.2. Objetivos específicos

Los objetivos específicos que nos hemos marcado son fomentar la idea de compartir coche y mas centralizado en el trayecto de casa al trabajo y viceversa, así conseguir un ahorro considerable tanto energético como económico, beneficios ambientales como la disminución de la polución o reducción de la contaminación acústica.

Otra consecuencia que se deriva de lo anterior es la eficacia en los desplazamientos y así mismo la posibilidad de la mejora de las relaciones sociales. Aunque el enfoque del proyecto esté centrado en los trayectos al trabajo, no se descarta la posibilidad de utilizarlo en viajes largos o en otro tipos de trayectos.

Otros de nuestros objetivos es que la gente coja conciencia que no es necesario la posesión

de un vehículo privado para poder realizar determinados trayectos, ya que nuestra aplicación es para usuarios que poseen coches y los que no. Por lo tanto otro objetivo implícito es que el usuario poseedor de un coche vaya cogiendo conciencia de que el uso del coche es para momentos puntuales, en los cuales podrá compartir su automóvil que supondrá un ahorro tanto para el poseedor del coche como para los pasajeros "anónimos".

Fomentar la utilización de aplicaciones Carpooling, es otro de nuestros objetivos, que pierdan el miedo a la utilización de estos recursos y así beneficiarse de todas las posibilidades que ofrece.

De todos estos objetivos se deriva que aunque una persona tenga coche no se vea obligada a utilizarlo para realizar un trayecto, incluso si en trayectos rutinarios durante la semana quiere optar por la opción de que unos determinados días quiera ir en su coche y otros en cambio no quiera conducir, el carpooling le permite esta posibilidad. De esta manera ponemos a disposición de la gente una forma de carpooling bastante flexible que se adapta a las necesidades del usuario.

3.3. Objetivos personales

El aprendizaje de nuevas tecnologías, que no se enseñan en la carrera, (lenguajes, API,. . .). Conseguir que la sociedad española cambie su idea sobre el Carpooling con una interface amena, sencilla y segura.

El aprendizaje de la documentación y formalización necesaria para la presentación de un proyecto, así como las diferentes baterías de pruebas para garantizar el buen funcionamiento de la aplicación.

Como último objetivo realizar algo útil para toda la sociedad y así aportar nuestro granito de arena por hacer de este mundo un sitio mas limpio y con menos contaminación.

3.4. Diagramas UML

Como lenguaje de modelado se ha elegido UML (Unified Modeling Language), que se trata del lenguaje de modelado de sistemas de software más conocido y usado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un plano del sistema (un modelo), que incluye aspectos conceptuales como los procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

3.5. Requisitos no funcionales

Hemos definido una serie de requisitos no funcionales que debe de cumplir nuestro proyecto. Un requisito no funcional es aquel que puede usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, que son los requisitos funcionales, que son aquellos que definen el comportamiento interno del software: algoritmos utilizados, detalles de la arquitectura del sistema, etc... Nuestros requisitos no funcionales en concreto son:

- Usabilidad : La interfaz de usuario ha de ser clara, intuitiva y rápida de usar. El uso del sistema no ha de requerir ningún tipo de aprendizaje previo para ninguno de los usuarios a los que está destinado.

- Accesibilidad: El sistema ha de ser accesible para cualquier navegador estándar actual. Se han de seguir al máximo posible los estándares de la web así como los criterios de buen diseño para personas con dificultades de acceso.

- Costo: Se ha procurado usar tecnologías que no impliquen un coste ni para nosotros ni para los usuarios.

- Mantenimiento y extensibilidad: El código fuente de la aplicación ha de ser lo más

fácil de mantener y extensible posible. El sistema ha de permitir la implementación de nuevas funcionalidades de forma en que no se tenga que rehacer la aplicación ni que pequeños cambios impliquen cambios traumáticos de gran calado en el sistema.

- Tecnologías estándar y código abierto: El sistema y código resultante se pretende que sea producto, en la medida de lo posible, del desarrollo con tecnologías de código abierto para que pueda ser fácilmente mantenido y modificado sin las restricciones que pudiera suponer en un futuro el uso, de por ejemplo, un SGBD propietario.

- Seguridad: El sistema ha de ser lo más seguro posible, para evitar el filtrado de datos personales, evitando ataques de inyección y validando los datos necesarios.

3.6. Diagrama de casos de uso

En UML, un diagrama de casos de uso es una especie de diagrama de comportamiento que define una notación gráfica para representar casos de uso. Los casos de uso se usan mucho para la captura de requisitos funcionales.

Un diagrama de casos de uso consta de actores, los cuales participan en alguno de los casos de uso del sistema. Los actores acostumbran a representar personas, pero también puede ser cualquier otro tipo de sistema como por ejemplo software de otros sistemas externos que interactúe con el sistema(como por ejemplo Google Maps).

En nuestro caso, nuestro diagrama de casos de uso consta de 5 actores:

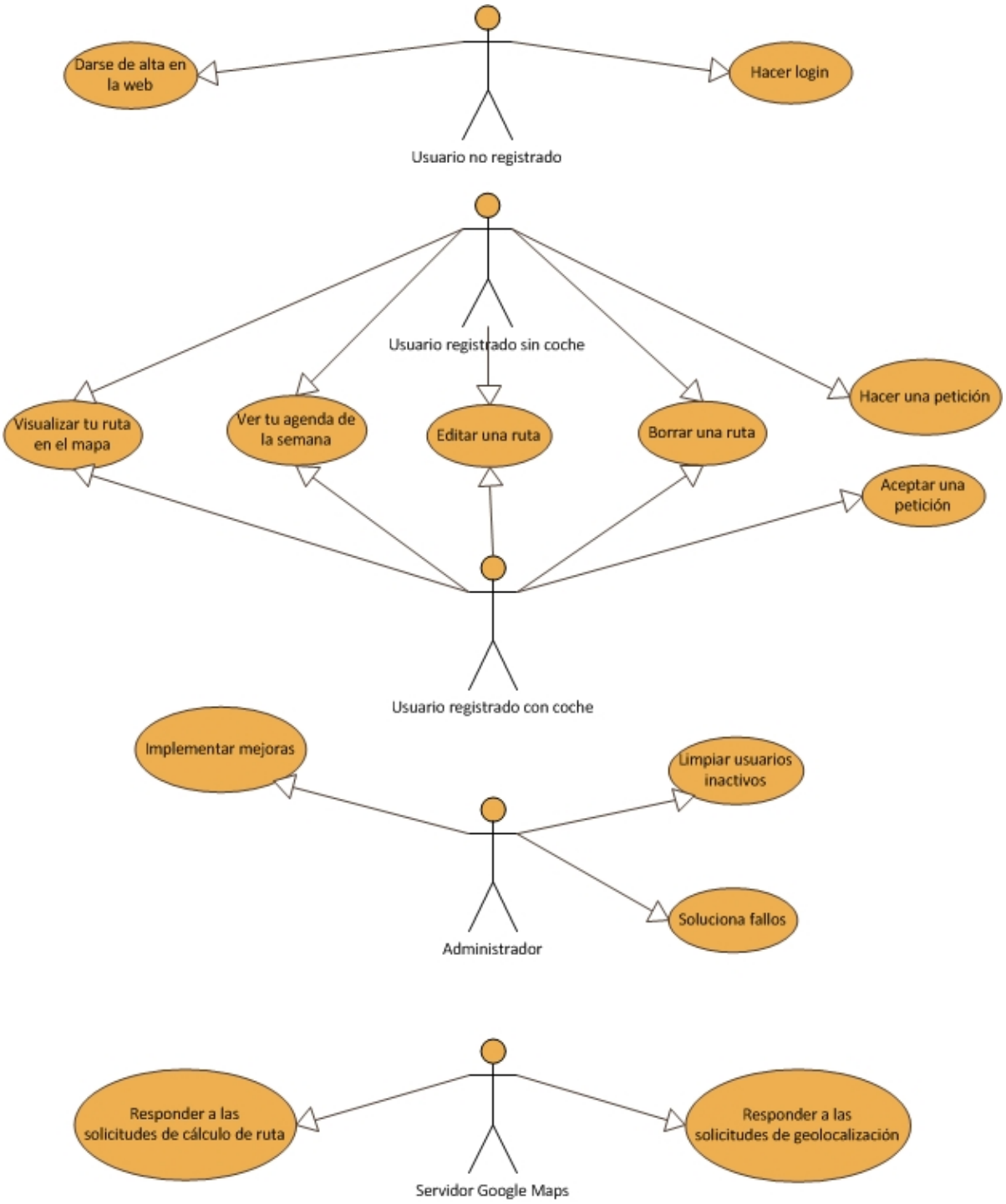
- Usuario no registrado en nuestra web

- Usuario registrado con coche en nuestra web.

- Usuario registrado sin coche en nuestra web.

- Administrador de la web

- Servidores de Google Maps


Darse de alta en la web
Hacer login
Usuario no registrado
Usuario registrado sin coche
Visualizar tu ruta en el mapa
Ver tu agenda de la semana
Editar una ruta
Borrar una ruta
Hacer una petición
Aceptar una petición
Usuario registrado con coche
Implementar mejoras
Limpiar usuarios inactivos
Soluciona fallos
Administrador
Responder a las solicitudes de cálculo de ruta
Responder a las solicitudes de geolocalización
Servidor Google Maps

3.7. Especificación de casos de uso

Un caso de uso es el conjunto de escenarios relacionados que describen de qué manera los actores usan el sistema para conseguir un determinado objetivo, y como actúa el sistema por detrás. Solo describiremos los casos de uso que consideramos más importantes:

Caso de uso: Registrarse en la web

Actores: Usuario no registrado

Precondición: Tener acceso a internet

Poscondición: Tener ya la posibilidad de acceder a nuestra web como usuario registrado

Escenario principal, curso lógico de sucesos:

Usuario	**Sistema**
1. Entra a nuestra web a través de la URL: pfcnacho.fdi.ucm.es	2.Carga la página principal de nuestra web
3. Clickea en el botón: Registrarse	4. Se carga la página de registro
5. El usuario rellana sus datos, y envía el formulario	6. Se introducen los valores en la base de datos, y el usuario estaría ya registrado. 7. Se le redirige al usuario a la página de inicio
8. El usuario entra con su nueva cuenta registrada	9. Se redirige al usuario a su página principal con el mapa

Escenario alternativo:

- Paso 5: Si el usuario al rellena los campos requeridos, rellena alguno de forma incorrecta, ya sea por que introduce caracteres inválidos, o por que el formato de la hora no es el correcto, se avisa al usuario del fallo, y se le deja en la misma página del formulario para que cambie los valores erróneos.

Caso de uso: Añadir una ruta

Actores: Usuario registrado con o sin coche; Servidor de Google Maps

Precondición: Estar registrado en la web

Poscondición: El usuario registrado dispondrá ya de una ruta registrada en el sistema,

con la que se podrá apuntar a una ruta con coche en caso de que no tenga coche, o podrá aceptar peticiones de ruta en caso de que tenga coche.

Escenario principal, curso lógico de sucesos:

Usuario	**Sistema**
1. El usuario pincha en la pestaña de:Añadir Ruta"	2.Carga la página de añadir ruta
3. El usuario rellana los datos de la ruta que quiere añadir, poniendo en Plazas Libres 0 en caso de que no tenga coche, y en caso de tenerlo, pondrá el número de plazas libres de las que dispone en su coche	4.El sistema lanza una petición al servidor de Google Maps, para que se dibuje en el mapa los datos de la ruta que está introduciendo el usuario
5. El usuario comprueba en el mapa el recorrido de su ruta, y en caso de estar todo correcto, pinchará sobre el botón Crear	6. Se guardan los datos en la base de datos

Escenario alternativo:

- Paso 3: Si el usuario al rellanar los campos requeridos, rellena alguno de forma incorrecta, ya sea por que introduce caracteres inválidos, o por que el formato de la hora no es el correcto, se avisa al usuario del fallo, y se le deja en la misma página del formulario para que cambie los valores erróneos.

- Paso 5: Si el usuario ve la ruta, y en este punto quiere cambiar algún dato introducido, aun esta a tiempo. En ese caso, se volvería al paso 3.

Caso de uso: Editar una ruta

Actores: Usuario registrado con o sin coche; Servidor de Google Maps

Precondición: Tener la ruta que se quiere editar creada ya en la base de datos

Postcondición: El usuario tendrá la ruta editada con los cambios que quería

Escenario principal, curso lógico de sucesos:

Escenario alternativo:

- Paso 5: Si el usuario al rellanar los campos requeridos, rellena alguno de forma incorrecta, ya sea por que introduce caracteres inválidos, o por que el formato de la hora

Usuario	Sistema
1. El usuario pincha en la pestaña de:Editar Ruta	2.Carga la página de editar ruta
3. El usuario selecciona en la agenda la ruta que quiere editar	4.El sistema carga en la página los datos de la ruta que quiere editar el usuario, mostrando en la pantalla la ruta en concreto tras hacer una petición al servidor de Google Maps.
5. El usuario modifica los datos que quiere editar	6. Se guardan los datos en la base de datos

no es el correcto, se avisa al usuario del fallo, y se le deja en la misma página del formulario para que cambie los valores erróneos.

- Paso 5: Si el usuario modifica los datos del origen o destino de la ruta, y le da al botón "Probar", se visualizará en el mapa la nueva ruta insertada, tras hacer el sistema una petición al servidor de Google Maps.

Caso de uso: Borrar una ruta

Actores: Usuario registrado con o sin coche; Servidor de Google Maps

Precondición: Tener la ruta que se quiere borrar creada ya en la base de datos

Postcondición: El usuario ya no dispondrá de esa ruta en la base de datos, así como todos los usuarios que estuvieran en esa ruta en caso de que fuera una ruta con coche.

Escenario principal, curso lógico de sucesos:

Usuario	Sistema
1. El usuario pincha en la pestaña de: Editar Ruta	2.Carga la página de editar ruta
3. El usuario selecciona en la agenda la ruta que quiere borrar	4.El sistema carga en la página los datos de la ruta que quiere editar el usuario, mostrando en la pantalla la ruta en concreto tras hacer una petición al servidor de Google Maps.
5. El usuario pincha sobre el botón Borrar Ruta	6.Se elimina la ruta de la base de datos, así como todas las rutas que contuviera en caso de que la ruta borrada fuera una ruta con coche

Caso de uso: Hacer una petición de ruta

Actores: Usuario registrado sin coche; Servidor de Google Maps

Precondición: Tener una ruta sin coche.

Postcondición: El usuario tendrá una petición pendiente a una ruta con coche.

Escenario principal, curso lógico de sucesos:

Usuario	**Sistema**
1. El usuario pincha en la pestaña de: Petición Ruta	2.Carga la página de petición ruta, cargando las posibles rutas a las que se puede apuntar el usuario.
3. El usuario pincha sobre la ruta que más le guste para hacer la petición	4.El sistema carga en la página como quedaría la ruta entera, gracias al algoritmo de filtrado explicado en la sección de algoritmos
5. El usuario pulsa sobre el botón Aceptar	6. Se guardan los datos en la base de datos como una nueva petición, que ya le llegará y aceptará o no el dueño de la ruta con coche

Escenario alternativo:

- Paso 5: Si el usuario al ver como queda la ruta, no le gusta, puede seleccionar otra ruta a la que pedir, y volvería al paso 3.

Caso de uso: Confirmar una petición de ruta

Actores: Usuario registrado con coche; Servidor de Google Maps

Precondición: Tener peticiones sobre alguna ruta propia.

Postcondición: El usuario tendrá un nuevo usuario en alguna ruta de las que tiene.

Escenario principal, curso lógico de sucesos:

Usuario	**Sistema**
1. El usuario pincha en la pestaña de: Confirmar Peticiones	2.Carga la página de confirmar peticiones, cargando las peticiones que tiene el usuario.
3. El usuario pincha sobre la ruta que más le guste para aceptar la petición	4.El sistema carga en la página como quedaría la ruta entera, gracias al algoritmo de filtrado explicado en la sección de algoritmos
5. El usuario pulsa sobre el botón Aceptar	6. Se guardan los datos en la base de datos , y se borra la petición

Escenario alternativo:

- Paso 5: Si el usuario al ver como queda la ruta, no le gusta, puede seleccionar otra ruta a la que pedir, y volvería al paso 3.

Capítulo 4

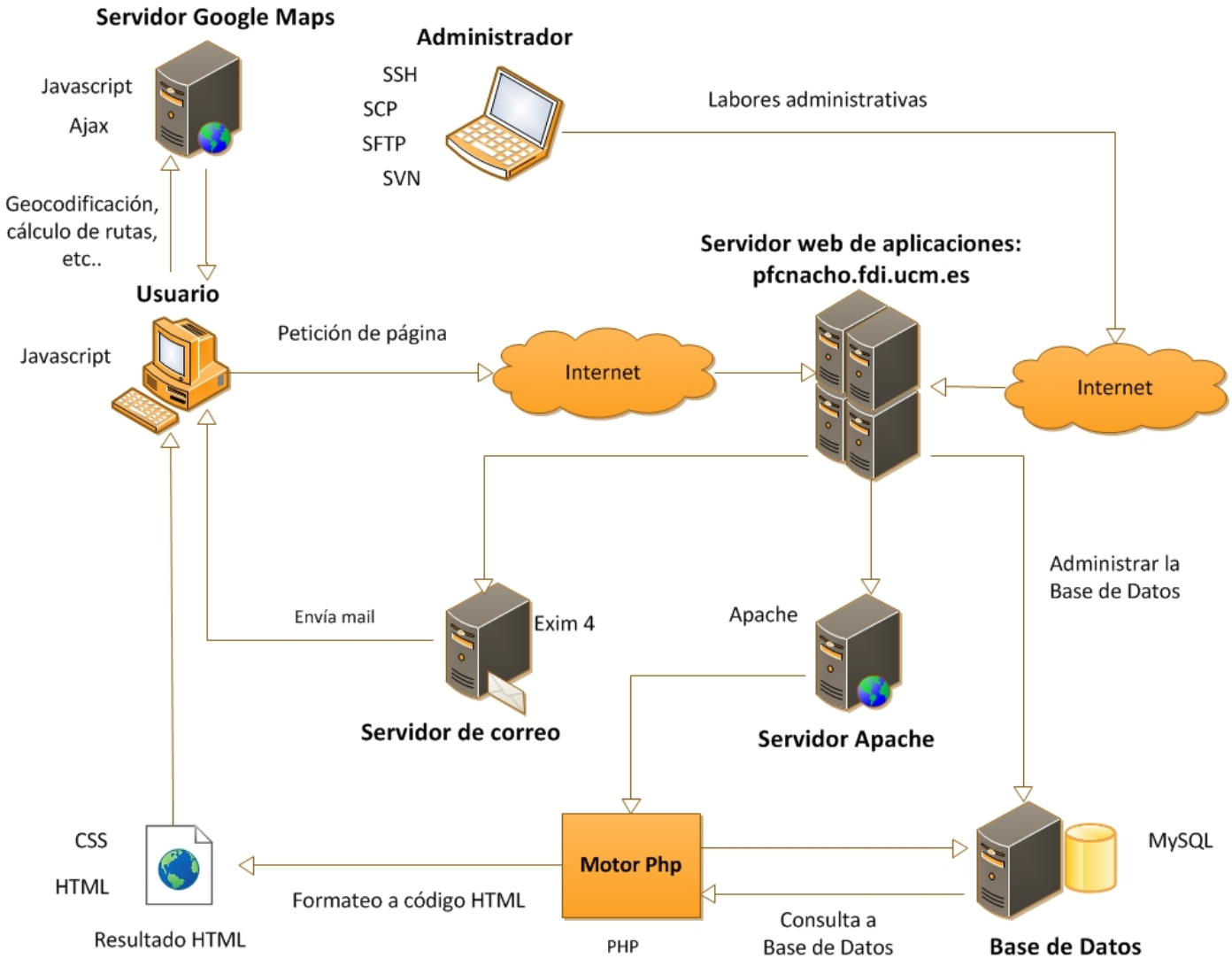
Diseño del Sistema

4.1. Arquitectura del Sistema

El sistema estará basado en una arquitectura cliente-servidor, en la que los usuarios del sistema se conectarán mediante el uso de un navegador web cualquiera.

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta.

La arquitectura de nuestro sistema, se puede ver en la siguiente ilustración, aunque se entrará en detalle de las tecnologías de cada módulo en el capítulo 5:



De forma abstracta, se pueden diferenciar 7 módulos distintos:

- **Módulo de Google Maps:** En este módulo se encuentra todo aquello que tiene que ver con la API de Google Maps, así como con el servidor de Google Maps al cual se le harán las peticiones para la geocodificación de puntos latitud/longitud, cáculo de rutas, visualización de mapas, etc.. El motor de la API de Google Maps será AJAX.

- **Módulo del Usuario:** Es el entorno desde donde el cliente accede a nuestra web, es decir, el navegador que usará, el cual tendrá que tener habilitado Javascript para su

ejecución.

- **Módulo del Administrador:** Son las tecnologías que empleamos los administradores para conectarnos al servidor y hacer las tareas oportunas. En concreto, usaremos SSH para conectarnos al servidor, así como SCP y SFTP para transferir archivos y colocarlos en el servidor. También dispondremos de un repositorio SVN donde iremos guardando las diferentes versiones de nuestro código.
- **Módulo Apache:** Apache será el servidor web que utilizaremos en el desarrollo de nuestro proyecto.
- **Módulo de Base de Datos:** Aquí será donde tendremos nuestra base de datos, la cual es una MySQL.
- **Módulo de Servidor de correo:** Exim4 será el servidor que usaremos para el envío de correos electrónicos a los usuarios de la web.
- **Módulo motor PHP:** Emplearemos PHP como lenguaje de programación en el lado del servidor web.

4.2. Patrones de diseño

Después de haber definido la arquitectura de nuestro sistema, entramos en la fase de diseño, donde modelamos el sistema en base a la arquitectura y a sus requerimientos. En concreto, los patrones de diseño que hemos empleado en la elaboración del proyecto son: Modelo Vista Controlador y el patrón Front Controller:

4.2.1. Modelo Vista Controlador

El patrón de diseño arquitectural del software Modelo Vista Controlador (MVC) [28] es uno de los más comunes y usados en aplicaciones web, debido a su idoneidad en este tipo de aplicaciones. Este patrón sigue un modelo que separa los datos de una aplicación, la interfaz

de usuario, y la lógica de control en tres componentes distintos. Estos 3 componentes se pueden asociar naturalmente: la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema Gestor de Base de Datos y la lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista y comunicar con el modelo. Pero veamos en detalle cada componente:

- Modelo: Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador, facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado. La lógica de los datos asegura la integridad de estos y permite derivar nuevos datos. En resumen, es la información almacenada en una base de datos o en XML, junto con las reglas de negocio que transforman esa información (teniendo en cuenta las acciones de los usuarios)

- Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario (la página HTML).

- Controlador: Este responde a eventos, usualmente acciones del usuario, reclama datos al modelo y los muestra en la la vista(código que obtiene datos dinámicamente y genera el contenido HTML)

Flujo de control del Modelo-Vista-Controlador:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario).

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista.

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

El Modelo-Vista-Controlador suele estar compuesto por varios modelos, varias vistas y varios controladores. Las vistas y los controladores suelen estar muy relacionados, ya que los controladores tratan los eventos que se producen en la interfaz gráfica(vista). La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre Vista y su correspondiente Controlador de eventos y acceso a datos. MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

Con este patrón de diseño obtenemos varias ventajas, claves en el desarrollo de aplicaciones web:

- La separación de la lógica de negocio y de la vista, que permiten la reusabilidad del modelo.

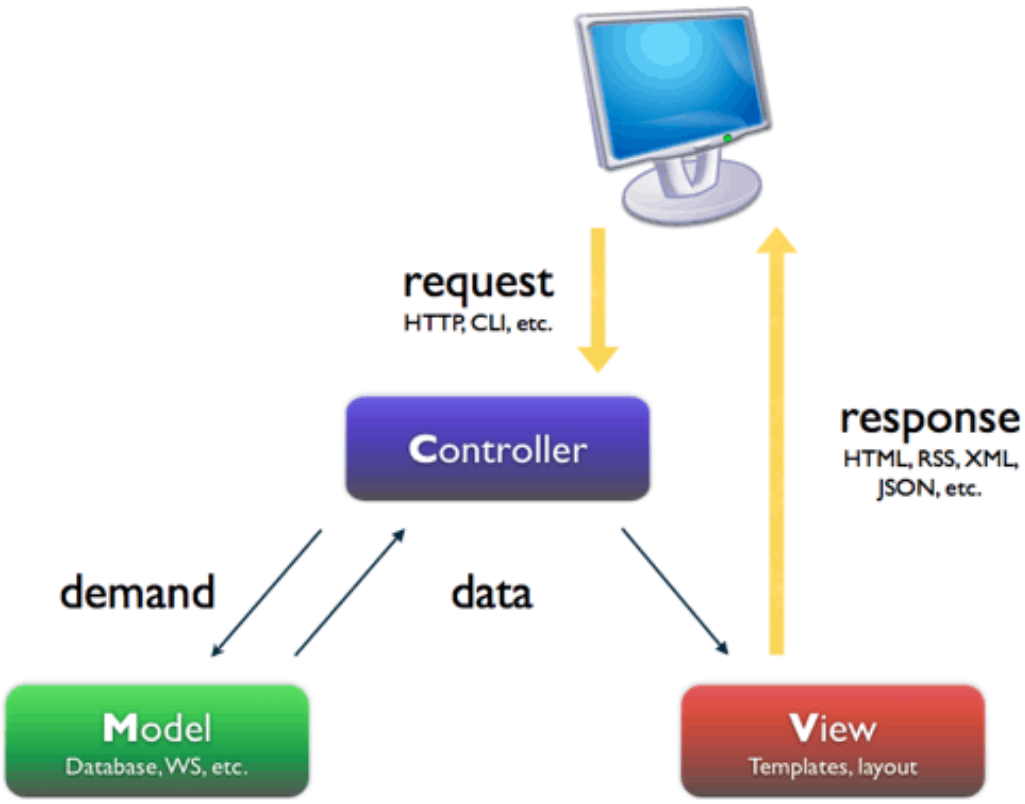
- Permite una división sencilla de roles, dejando el diseño de la vista con la mínima cantidad de código de programación posible.

- Simplicidad en el mantenimiento de la web, y más escalable.

Sin embargo y como todo, este patrón presenta también inconvenientes que hemos tenido en cuenta a la hora de tomar la decisión de usar o no el MVC, pero que finalmente han ganado las ventajas. Algunos inconvenientes son:

- Tener que ceñirse a una estructura predefinida, lo que a veces puede incrementar la complejidad del sistema.
- La curva de aprendizaje para los nuevos desarrolladores se estima mayor que la de los modelos más simples.
- La distribución de componentes obliga a crear y mantener un mayor número de ficheros.

En la siguiente ilustración se puede ver la arquitectura del MVC [32]:



4.2.2. Patrón Front Controller

Usamos el patrón Front Controller [30] para facilitar la creación de un punto de interacción inicial entre la interfaz de usuario y el resto del sistema que gestione todas las solicitudes. Con este patrón tenemos un objeto que acepta todos los requerimientos de un cliente y los direcciona a los controladores apropiados y así nos evita tener que repetir la misma lógica de control en diversos puntos y la agrupa en un único punto de entrada.

En nuestro caso, usamos la página index.php para gestionar todas las peticiones y mostrar el resultado en base a ellas. Por ejemplo, todas las páginas de nuestra web serán de la forma:

- http://pfcnacho.fdi.ucm.es/index.php?controlador=request
- http://pfcnacho.fdi.ucm.es/index.php?controlador=add
- http://pfcnacho.fdi.ucm.es/index.php?controlador=request&accion=possib
- etc..

El fichero index.php contendrá únicamente el siguiente código:

```
1 <?php
2 //Incluimos el FrontController
3 require 'libs/FrontController.php';
4 //Lo iniciamos con su método estático main.
5 FrontController::main();
6 ?>
```

Será está función FrontController::main la que lea el controlador que se le pasa a la página como parámetro, e inicialize el controlador correspondiente. Si se le pasa también alguna acción, será aquí donde se llame a la función que la realiza. Se puede ver en el siguiente código.

```
1 <?php
2 class FrontController
3 {
4   static function main()
5   {
6     require 'libs/Config.php'; //de configuracion
```

```
7       require 'libs/SPDO.php'; //PDO con singleton
8       require 'libs/View.php'; //Mini motor de plantillas
9
10      require 'config.php'; //Archivo con configuraciones.
11
12      //Formamos el nombre del Controlador o en su defecto, tomamos que es el
            IndexController
13      if(! empty($_GET['controlador']))
14              $controllerName = $_GET['controlador'] . 'Controller';
15      else
16              $controllerName = "indexController";
17
18      //Lo mismo sucede con las acciones, si no hay accion, tomamos index como
            accion
19      if(! empty($_GET['accion']))
20              $actionName = $_GET['accion'];
21      else
22              $actionName = "index";
23
24      $controllerPath = $config->get('controllersFolder') . $controllerName .
            '.php';
25
26      //Incluimos el fichero que contiene nuestra clase controladora solicitada
27      if(is_file($controllerPath))
28              require "$controllerPath" ;
29      else
30              die('El controlador no existe - 404 not found '.$controllerPath);
31
32      //Si no existe la clase que buscamos y su acción, tiramos un error 404
33      if (is_callable(array($controllerName, $actionName)) == false)
34      {
35        trigger_error ($controllerName . '->' . $actionName . '` no existe',
              E_USER_NOTICE);
36        return false;
37      }
38      //Si todo esta bien, creamos una instancia del controlador y llamamos a
            la accion
39      $controller = new $controllerName();
40      $controller->$actionName();
41
42    }
43  }
44  ?>
```

4.3. Metodología de programación

La metodología que hemos llevado a cabo ha sido la programación extrema. Esta consiste en la estrecha colaboración entre el cliente y el creador y la comunicación cara a cara.

Primeramente se fija una versión sencilla de la herramienta con las mínimas funciones y un desarrollo básico para que, posteriormente, mediante reuniones periódicas, se vaya probando lo realizado y mejorando con nuevas opciones. Por tanto, el producto se va adaptando sobre la marcha a las necesidades del cliente, dando el producto por finalizado cuando se alcanza lo que el cliente pretendía en un principio.

Los valores originales de la programación extrema son: simplicidad, comunicación, retroalimentación (feedback) y coraje. Un quinto valor, respeto, fue añadido en la segunda edición de [1]. Los cinco valores se detallan a continuación:

- **Simplicidad:** La simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento, ya que un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hacen que la complejidad aumente exponencialmente. También se aplica la simplicidad a la documentación, de tal forma que el código debe comentarse en su justa medida, intentando eso sí, que el codigo esté autodocumentado, ya sea eligiendo nombres de variables, métodos o clase adecuados. Aplicando la simplicidad junto con la autoría colectiva del código y la programación por parejas, se asegura que cuanto más grande se haga el proyecto, todo el equipo conocerá más y mejor el sistema completo.

- **Comunicación:** Para los programadores el código comunica mejor cuanto más simple sea, debido a que si el código es complejo hay que esforzarse para hacerlo inteligible. Debe comentarse sólo aquello que no va a variar, por ejemplo el objetivo de una clase o la funcionalidad de un método. Las pruebas unitarias son otra forma de comunicación, ya que describen el diseño de las clases y los métodos al mostrar ejemplos concretos de como utilizar su funcionalidad. Los programadores se comunican constantemente gracias a la programación por parejas. La comunicación con el cliente es fluida ya que el cliente forma parte del equipo de desarrollo. El cliente decide que características tienen prioridad y siempre debe estar disponible para solucionar dudas.

- **Retroalimentación (feedback):** Al tener al cliente integrado en el equipo de desarrollo, su opinión sobre el estado del proyecto se conoce en tiempo real. Al realizarse ciclos muy cortos, tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo que es más importante. El código también es una fuente de retroalimentación gracias a las herramientas de desarrollo. Por ejemplo, las pruebas unitarias informan sobre el estado de salud del código. Ejecutar las pruebas unitarias permite descubrir fallos debidos a cambios recientes en el código.

- **Valentía o Coraje:** Este punto puede llegar a parecer fuera de lugar comparando con los anteriores, pero es un atributo que deben tener los gerentes a la hora de aceptar la programación en parejas, ya que pueden pensar que la productividad vaya a reducirse a la mitad al estar sólo la mitad de los programadores escribiendo código. Por tanto hay que ser valiente para confiar en que la programación por parejas beneficia la calidad del código sin repercutir negativamente en la productividad. La valentía también le permite a los desarrolladores que se sientan cómodos con reconstruir su código cuando sea necesario. Esto significa revisar el sistema existente y modificarlo si con ello los cambios futuros se implementarán más fácilmente. Otro ejemplo de valentía es saber cuando desechar un código: valentía para quitar código fuente obsoleto, sin importar cuanto esfuerzo y tiempo se invirtió en crear ese códio. Además, valenía significa persistencia: un programador puede permanecer sin avanzar en un problema complejo un día entero, y luego resolverá asiduamente al día siguiente, sólo si es persistente.

- **Respeto:** Tanto entre programadores (con una serie de normas en el desarrollo del código) como entre los creadores y el cliente.

Además de los valores, la programación extrema debe cumplir con una serie de características:

- **Desarrollo iterativo e incremental**: pequeñas mejoras, unas tras otras.

- **Pruebas unitarias** continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión, que son aquellas que intentan descubrir las causas de nuevos errores, carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software, incluido por cambios recientemente realizados en parte de la aplicación que anteriormente al citado cambio no eran propensas a este tipo de error.

- **Programación en parejas**: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera (el código es revisado y discutido mientras se escribe) es más importante que la posible pérdida de productividad inmediata.

- Frecuente **integración del equipo de programación con el cliente** o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.

- **Corrección** de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.

- **Refactorización del código**, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha producido ningún fallo.

- **Propiedad del código compartido**: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- **Simplicidad en el código**: la programación extrema apuesta por realizar algo simple y más adelante cambiarlo si se requiere antes que realizar algo complejo y quizás no utilizarlo nunca.

4.3.1. Adaptación de la metodología empleada al proyecto en curso

A continuación se detalla el proceso realizado mediante Programación Extrema adaptado particularmente a este proyecto:

Este trabajo ha sufrido un **desarrollo iterativo e incremental**, debido a que según se iban alcanzando objetivos previamente fijados se iban proponiendo nuevas ideas y funcionalidades que añadir al proyecto, de manera que se ha evolucionado en el desarrollo de algo básico a lo que se propuso como objetivo final.

A lo largo del desarrollo del proyecto, se han realizado diversas **pruebas unitarias**, ya que se ha probado el correcto funcionamiento para cada nueva mejora que se ha ido incluyendo al producto final. Estas pruebas no son automáticas, ya que en una página web es más complicado de automatizar. Por tanto, una característica implícita a lo anterior que se ha llevado a cabo en la realización de este proyecto es la **corrección de errores** antes de añadir nueva funcionalidad.

En este caso particular, al tratarse de un proyecto de fin de carrera, los tutores toman el papel del cliente, ya que son ellos quienes marcan los requisitos del proyecto, imponen los objetivos y modifican las distintas versiones del producto adaptando las nuevas funcionalidades de la página web a los objetivos fijados en un principio. Por tanto la **integración del equipo de programación con el cliente** es total, realizándose pequeñas entregas a través de todo el proceso y reuniones periódicas entre nosotros y los tutores.

La **programacioń en pareja**, si bien es cierto que siendo un grupo de 3 no siempre ha sido posible, hemos tendido a trabajar en equipo y que todos conozcamos los cambios que está realizando el compañero, ayudándole si fuera preciso. La **propiedad compartida del código**, es por tanto, otra característica que se cumple. También se realiza una **refactori-**

zación constante del proyecto, debido a que se actualiza y mejora la web en distintas fases del proyecto, sin que esto produzca un retroceso en el código ya implementado.

Por último, aunque no por ello menos importante, se debe mencionar que el diseño cumple con la característica referente a la **simplicidad en el código**, ya que al haber seguido un desarrollo incremental, se ha partido de un código simple para posteriormente adaptarlo a los objetivos finales, de manera que no se ha realizado nada demasiado complejo que finalmente no haya sido utilizado, a pesar de que usamos patrones como el MVC entre otros, que aunque parezca que complica el código, es algo que mejora la calidad de la web.

4.3.2. Iteraciones

En este apartado se expondrá una división y planificación orientativa del trabajo realizado a modo de cuaderno de bitácora organizado en forma de hitos, que se han ido realizando y exponiendo durante nuestras reuniones con los profesores del proyecto. Durante el resto de la sección se hablará de cliente cuando se quiera referir a los tutores de este PFC.

Hito 1: Estudio de la API de Google Maps

Como primer objetivo del proyecto, nos proponemos el estudio de la API de Google Maps, así como ir haciendo las primeras pruebas para calcular la ruta entre dos puntos, y como se incrusta un mapa de Google Maps en una web.

Hito 2: Estudio de diferentes tecnologías webs

En principio estudiamos tecnologías de interfaz web como Ajax o Ruby on Rails, así como gestores de contenido de alto nivel como Drupal o Joomla, pero nos decantamos por escribir nosotros la web desde cero. Para ello, usamos como software de diseño web el Dreamweaver CS5.5, usando javascript y php. Para la página web, decidimos usar una Mysql, y como servidor el que nos facilitan los tutores, que en este caso es un Apache. Para el trabajo en

local, usaremos XAMPP.

Todas estas decisiones están mejor detalladas en la sección 7. Tecnologías.

Hito 3: Estudio de posibles ideas para el diseño gráfico de la web

Realizamos un pequeño esquema de las tres pantallas principales del portal, con un posible diseño. En concreto, lo hacemos de:

- Página inicial
- Página de introducción de datos
- Página de búsqueda

Hito 4: Diseño de la Base de datos

Hacemos un primer diseño de la base de datos, para poder empezar a guardar usuarios y rutas. A medida que avanza en complejidad el proyecto, se van modificando y añadiendo nuevas tablas. El diseõ final de la base de datos se puede ver en el anexo 9.3. Diagrama Bases de Datos.

Hito 5: Realización de una interfaz web en HTML sencilla

Hacemos un primer boceto de la página, sin demasiada funcionalidad todavía, pero que nos sirve de base para ir añadiendo el resto de mejoras y requisitos de nuestro sistema.

Hito 6: Creación de rutas sencillas funcionando (dos usuarios)

Ya se muestra como un usuario puede recoger a otro usuario.

En este punto nos damos cuenta que necesitamos un algoritmo de enrutamiento mucho mas complejo del que tenemos, para poder recoger a más usuarios

Hito 7: Establecer el protocolo de comunicación entre usuarios

Llegamos a la conclusión de que el mejor modo para que los usuarios se comuniquen entre si es a través del correo electrónico.

Hito 8: Añadir a la interfaz web el poder establecer un umbral de tiempo

El usuario a partir de ahora podrá establecer cuanto es de flexible su tiempo en el origen y destino de su ruta. A la hora de pedir una ruta, solo se mostrarán rutas que encajen dentro del umbral de tiempo.

Hito 9: Creación de rutas complejas, con un conductor y más de dos ocupantes

Realizamos el algoritmo de filtrado, que se detalla en la sección 6.1. Algoritmo de cálculo de ruta.

Hito 10: Creación de la interfaz para el dueño del coche

En esta interfaz, el dueño del coche debe poder confirmar que acepta a un nuevo usuario en su coche para una determinada ruta, enviándole un correo al otro usuario confirmando que es aceptado en el coche.

Hito 11: Mejora del algoritmo de filtrado antes de usar el API de Google Maps

A la hora de hacer peticiones, filtramos los posibles ocupantes mediante la latitud/longitud para evitar consultas de Google. En concreto, solo podrá recoger aquellos puntos que se encuentren dentro del rectángulo delimitado por el origen y destino del trayecto. También se tenedrá en cuenta el ángulo y la hora. Todo esto está mejor explicado en la sección 6.2 Algoritmo de filtrado.

Hito 12: Creamos nuestro logo de QueTeCunda

Diseñamos el logo de nuestra web, y elegimos el naranja como color predominante del sistema:

Hito 13: Se mejora la página principal de la web

Se mejora la página principal de la web, cambiando la localización del login y password, y dotándole de un mejor diseño.

Hito 14: Se termina de mejorar por completo la interfaz de la web

Ya se termina de mejorar la interfaz de la web. Para ver como termina, véase la guía de la web en el 9. Anexo.

Hito 15: Se mejora el mail que se envía a los usuarios

Se mejora el mail que se envían a los usuarios.

Capítulo 5

Implementación de los módulos principales

En esta sección detallaremos las tecnologías que hemos usado en nuestro proyecto, así como las alternativas y la justificación de la elección. Destacamos a la API de Google Maps sobre el resto, ya que es el centro de nuestro sistema. Tambíen comentaremos los programas que hemos utilizado en el desarrollo del proyecto.

5.1. Módulo Google Maps

Google Maps [4] es un servidor de aplicaciones de mapas en la Web. Ofrece imágenes de mapas desplazables, así como fotos satelitales del mundo e incluso la ruta entre diferentes ubicaciones o imágenes a pie de calle. Ofrece, asimismo, la posibilidad de que cualquier propietario de una página Web integre muchas de sus características a su sitio, lo cual se adapta muy bien a las necesidades de nuestro proyecto.

Haremos uso de la versión 3 de la Google Maps JavaScript API, ya que la versión 2 de esta API quedó descartada oficialmente. Google Maps JavaScript API permite insertar Google Maps en tus páginas web. La versión 3 de esta API está especialmente diseñada para proporcionar una mayor velocidad y que se pueda aplicar más fácilmente tanto a móviles como a las aplicaciones de navegador de escritorio tradicionales.

El API proporciona diversas utilidades para manipular mapas y para añadir contenido

al mapa mediante diversos servicios, permitiéndote crear sólidas aplicaciones de mapas en tu sitio web.

Destacar, que el 27 de Octubre de 2011, ya con nuestro proyecto iniciado, Google anunció el fin del "uso gratis ilimitado" de su API de Google Maps introduciendo el límite de 25.000 mapas básicos o 2.500 mapas estilizados por día, a partir del cual habrá que pagar para que nuestra aplicación puedan seguir funcionando. Disponemos de alternativas viables como Open Street Map o MapQuest, aunque el coste de Google Maps parece bastante asumible ya que disponemos, en caso de mapas simples, de paquetes de 25.000 unidades por 4 dólares.

Google Maps fue una de las primeras aplicaciones basadas en AJAX de uso masivo por parte de los usuarios.

5.1.1. AJAX

AJAX [5], acrónimo de Asynchronous JavaScript And XML, es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML. [6]

AJAX en nuestro proyecto

Sin tener en cuenta el AJAX como motor de la API de Google Maps, no lo hemos tenido en cuenta para el desarrollo de nuestro proyecto por las siguientes causas:

- Las páginas con AJAX son más difíciles de desarrollar que las páginas estáticas, y en

nuestro caso salvo la componente de Google Maps, no era preciso un recargo dinámico de la página.

- Las páginas creadas dinámicamente mediante peticiones sucesivas AJAX, no son registradas de forma automática en el historial del navegador, así que haciendo clic en el botón de "volver"del navegador, el usuario no será devuelto a un estado anterior de la página, en cambio puede volver a la última página que visitó. Soluciones incluyen el uso de IFrames invisible para desencadenar cambios en el historial del navegador y el cambio de la porción de anclaje de la dirección.

- Los motores de búsquedas no entienden JavaScript. La información en la página dinámica no se almacena en los registros del buscador.

- Hay problemas usando Ajax entre nombres de dominios. Eso es una función de seguridad.

- El sitio con Ajax usa más recursos en el servidor. Recomendación: sólo usar las peticiones necesarias en AJAX, no desarrollar todo el sitio en AJAX. Con esto garantizamos menos recursos del servidor.

- Es posible que páginas con Ajax no puedan funcionar en teléfonos móviles, PDA u otros aparatos. AJAX no es compatible con todos los software para ciegos u otras discapacidades.

5.1.2. API de Google Maps

En este apartado presentaremos los servicios más importantes que nos ofrece la API de Google Maps y que usamos en nuestro proyecto. [7]

Geocodificador

La geocodificación es el proceso de convertir las direcciones en modo texto en las coordenadas geográficas (latitud/longitud), que se pueden utilizar para colocar marcadores o la

posición del mapa. La API de Google Maps proporciona una clase geocodificador para la geocodificación a partir e los datos de entrada del usuario.

Solicitudes de geocodificación

El acceso al servicio de geocodificación se realiza de forma asíncrona, ya que la API de Google Maps necesita hacer una llamada a un servidor externo. Por esa razón, se necesita pasar un método de devolución de llamada para que se ejecute tras la finalización de la solicitud. Este método de devolución procesa el resultado. Hay que tener en cuenta que el geocodificador puede devolver más de un resultado.

Se accede al servicio de Google Maps API de geocodificación dentro de su código a través del objeto google.maps.Geocoder. El método Geocoder.geocode () inicia una solicitud para el servicio de geocodificación, pasándole un objeto GeocodeRequest, que contiene los términos de entrada y un método de devolución de llamada para ejecutar la recepción de la respuesta.

El objeto GeocodeRequest literal contiene los siguientes campos:

- address: string, (dirección que se desea geocodificar)
- latLng: LatLng, (latitud/longitud para el que desea obtener el más cercano punto con dirección legible)
- bounds: LatLngBounds, (los límites para los que se desea usar. Es opcional)
- region: string (especifica el código de la región. Es opcional también)

Se puede pasar tanto una dirección o una latitud/longitud (en este caso de pasar una latitud/longitud, el geocodificador realiza lo que se conoce una codificación geográfica inversa.)

Respuestas de geocodificación

El servicio de geocodificación requiere un método de devolución de llamada como ya hemos explicado, para ejecutar en la devolución de los resultados del geocodificador. Esta

devolución de llamada debe pasar dos parámetros para contener los resultados y un código de estado, en ese orden. Si el geocodificador devuelve más de una entrada, el objeto GeocoderResults será un array.

El objeto GeocoderResults representa un resultado de Geocodificación único y es un objeto de la forma siguiente:

```
1
2  results[]: {
3  types[]: string,
4  formatted address: string,
5  address components[ ]: {
6    short name: string,
7    long name: string,
8    types[]: string
9  },
10 geometry: {
11   location: LatLng,
12   location type: GeocoderLocationType
13   viewport: LatLngBounds,
14   bounds: LatLngBounds
15 }
16}
```

donde:

- types[]: es una matriz que indica el tipo del resultado obtenido. Esta matriz contiene un conjunto de uno o más etiquetas que identifican el tipo de función que devuelve en el resultado. Por ejemplo, un código geográfico de Chicago devuelve "localidad", que indica que Chicago es una ciudad, y también devuelve "político"que indica que es una entidad política.

- formatted address: es una cadena que contiene la dirección legible de este lugar. A menudo, esta dirección es equivalente a la "dirección postal".

- address components[ ]: es una matriz que contiene los componentes de la dirección por separado, como se explicó anteriormente.

- location: contiene la latitud/longitud geocodificada.

- location type: almacena los datos adicionales acerca de la ubicación especificada.

Ejemplo de geocodificación

```
1   var geocoder;
2   var map;
3   function initialize() {
4     geocoder = new google.maps.Geocoder();
5     var latlng = new google.maps.LatLng(-34.397, 150.644);
6     var myOptions = {
7       zoom: 8,
8       center: latlng,
9       mapTypeId: google.maps.MapTypeId.ROADMAP
10    }
11    map = new google.maps.Map(document.getElementById("map_canvas"),
          myOptions);
12  }
13
14  function codeAddress() {
15    var address = document.getElementById("address").value;
16    geocoder.geocode( { 'address': address}, function(results, status) {
17      if (status == google.maps.GeocoderStatus.OK) {
18        map.setCenter(results[0].geometry.location);
19        var marker = new google.maps.Marker({
20            map: map,
21            position: results[0].geometry.location
22        });
23      } else {
24        alert("Geocode was not successful for the following reason: " +
              status);
25      }
26    });
27  }
28
29<body onload="initialize()">
30 <div id="map_canvas" style="width: 320px; height: 480px;"></div>
31  <div>
32    <input id="address" type="textbox" value="Madrid, Spain">
33    <input type="button" value="Encode" onclick="codeAddress()">
34  </div>
35</body>
36
37}
```

Ejemplo de geocodificación inversa

Recordamos que la geocodificación inversa es la traducción de una ubicación en el mapa en una dirección legible por humanos. El siguiente ejemplo geocodifica un valor latitud/lon-

gitud y centra el mapa en ese lugar:

```
1  var geocoder;
2  var map;
3  var infowindow = new google.maps.InfoWindow();
4  var marker;
5  function initialize() {
6    geocoder = new google.maps.Geocoder();
7    var latlng = new google.maps.LatLng(40.730885,-73.997383);
8    var myOptions = {
9      zoom: 8,
10     center: latlng,
11     mapTypeId: google.maps.MapTypeId.ROADMAP
12   }
13   map = new google.maps.Map(document.getElementById("map_canvas"),
         myOptions);
14 }
15
16 function codeLatLng() {
17   var input = document.getElementById("latlng").value;
18   var latlngStr = input.split(",",2);
19   var lat = parseFloat(latlngStr[0]);
20   var lng = parseFloat(latlngStr[1]);
21   var latlng = new google.maps.LatLng(lat, lng);
22   geocoder.geocode({'latLng': latlng}, function(results, status) {
23     if (status == google.maps.GeocoderStatus.OK) {
24       if (results[1]) {
25         map.setZoom(11);
26         marker = new google.maps.Marker({
27             position: latlng,
28             map: map
29         });
30         infowindow.setContent(results[1].formatted_address);
31         infowindow.open(map, marker);
32       }
33     } else {
34       alert("Geocoder failed due to: " + status);
35     }
36   });
37 }
```

Calculador de rutas

Se pueden calcular rutas, usando distintos medios de transporte, entre 2 puntos usando el objeto DirectionsService. Este objeto se comunica con el servicio de la API de Google Maps, que recibe las peticiones de calcular rutas y devuelve las rutas calculadas. Se pueden especificar para las direcciones tanto direcciones postales como valores de latitud/longitud.

Veamos las características más importantes de este servicio:

Petición de calcular ruta

El acceso al servicio se realiza de forma asíncrona, ya que la API de Google Maps necesita hacer una llamada a un servidor externo. Por esa razón, se necesita pasar un método de devolución de llamada que se ejecutará tras la finalización de la solicitud. Este método de devolución de llamada debe procesar el resultado. El servicio puede llegar a devolver más de un posible itinerario como un conjunto de rutas separadas [ ].

Para ello, se crea un objeto DirectionsService y un método DirectionsService.route call () para iniciar una solicitud al servicio de calculo de rutas, pasándole un objeto Directions-Request que contiene los términos de entrada y un método de devolución de llamada para ejecutarse tras la finalización de la solicitud..

El objeto DirectionsRequest contiene los siguientes campos:

- origin: LatLng | String, (especifica la ubicación de inicio para el cálculo de la ruta)
- destination: LatLng | String, (especifica la ubicación de destino para el cálculo de la ruta)
- travelMode: TravelMode, (especifica qué modo de transporte se va a utilizar en el cálculo de las direcciones.)
- unitSystem: UnitSystem, (especifica qué sistema de unidades a utilizar para mostrar los resultados. METRIC mostrará la distancia en kilómetros, e IMPERIAL en millas)
- waypoints[ ]: DirectionsWaypoint, (especifica los puntos intermedios por donde tiene que pasar la ruta)
- optimizeWaypoints: Boolean, (especifica que la ruta con los puntos de referencia suministrados será la óptima o no. Si está a true, el servicio devolverá los puntos de interés ordenados en un campo waypoint order)

- provideRouteAlternatives: Boolean, (Si esta a true, espeficica rutas alternativas en la respuesta)
- avoidHighways: Boolean, (Si está a true, indica que en la ruta calculada deben evitarse carreteras principales siempre que sea posible)
- avoidTolls: Boolean (Si está a true, indica que en la ruta calculada deben evitarse carreteras de peaje siempre que sea posible)
- region: String (espeficica el código de región)

Resultado del cálculo de la ruta

El objeto DirectionsResult contiene el resultado de la consulta del cáculo de una ruta, para usarlo nosotros directamente o para pasárselo a un objeto DirectionsRenderer, que será el encargado de dibujar la nueva ruta calculada en el mapa.

Para mostrar un DirectionsResult utilizando un DirectionsRenderer, hay que realizar los siguientes pasos:

1. Crear un objeto DirectionsRenderer.
2. Llamar a setMap () para enlazar con el mapa pasado.
3. Llamar a setDirections (), pasándole el DirectionsResult como se señaló anteriormente. Se detectará automáticamente cualquier cambio en sus propiedades y actualizará el mapa cuando sus direcciones asociadas hayan cambiado.

Ejemplo del cálculo de la ruta

Podemos ver un ejemplo de cálculo de ruta entre 2 puntos en el siguiente código, que por simplificar solo contendrá el código necesario para generar la ruta, dando por hecho que el usuario seleccionará el origen y el destino de la ruta aunque no se muestre como:

```
1 var directionsDisplay;
2 var directionsService = new google.maps.DirectionsService();
3 var map;
```

```
4
5  function initialize() {
6    directionsDisplay = new google.maps.DirectionsRenderer();
7    var chicago = new google.maps.LatLng(41.850033, -87.6500523);
8    var myOptions = {
9      zoom:7,
10     mapTypeId: google.maps.MapTypeId.ROADMAP,
11     center: chicago
12   }
13   map = new google.maps.Map(document.getElementById("map_canvas"), myOptions)
         ;
14   directionsDisplay.setMap(map);
15 }
16
17 function calcRoute() {
18   var start = document.getElementById("start").value;
19   var end = document.getElementById("end").value;
20   var request = {
21     origin:start,
22     destination:end,
23     travelMode: google.maps.TravelMode.DRIVING
24   };
25   directionsService.route(request, function(result, status) {
26     if (status == google.maps.DirectionsStatus.OK) {
27       directionsDisplay.setDirections(result);
28     }
29   });
30 }
```

Objetos DirectionsResult, DirectionsRoute, DirectionsLeg y DirectionsStep

Al enviar una solicitud al DirectionsService, se recibirá una respuesta que consiste en un código de estado y un resultado que será un objeto DirectionsResult, el cual únicamente consta de un solo campo:

- routes [ ] : contiene un conjunto de objetos DirectionsRoute. Cada ruta indica una manera de llegar desde el origen hasta el destino previsto en el DirectionsRequest. Por lo general, sólo una ruta se devuelve por solicitud, a menos que el campo de la solicitud provideRouteAlternatives se establece en true, y se devuelvan rutas alternativas por tanto.

Un DirectionsRoute contiene un único resultado a partir del origen y de destino especificados. Este camino puede constar de uno o más puntos intermedios dependiendo de si han

sido especificados o no los waypoints (serán del tipo DirectionsLeg). Un objeto DirectionsRoute consta de los siguientes campos:

- legs[ ]: contiene un array de objetos de tipo DirectionsLeg, cada uno de los cuales contiene información sobre un tramo de la carretera entre puntos especificados en los waypoints. Un camino sin puntos de referencia o waypoints contendrá exactamente un DirectionsLeg. Cada pata se compone de una serie de DirectionSteps.
- waypoint order: contiene un array con el orden de los waypoints de la ruta calculada. Esta lista contendrá un orden alterado si en la DirectionsRequest, el parámetro optimizeWaypoints estaba a true.
- overview path: contiene un array de latitudes/longitudes que representan aproximadamente el resultado de la ruta calculada.
- bound: indica los límites de latitud/longitud de la ruta calculada.
- copyrights: contiene el texto de derechos de autor que se muestra para esta ruta.
- warnings[ ]: muestra advertencias al verificar direcciones.

Un DirectionsLeg para rutas que no contienen puntos de referencia, define una sola "pata"de un viaje desde el origen hasta el destino en el camino calculado. Si la ruta tiene uno o más puntos de referencia, la ruta consistirá en una o más patas.

El objeto DirectionsLeg consta de los siguientes campos:

- steps[ ]: contiene un array de objetos DirectionsStep, que denotan la información sobre cada paso por separado de la pata del viaje.
- distance: determinar la distancia total cubierta de este tramo.
- duration: determinar la duración total cubierta de este tramo.
- start location: contiene la latitud/longitud del origen de este tramo.

- end location: contiene la latitud/longitud del destino de este tramo.
- start address: contiene la codificación postal del origen de este tramo.
- end address: contiene la codificación postal del destino de este tramo.

Un DirectionsStep es la unidad más atómica de la descripción de una ruta, que contiene paso a paso instrucciones específicas de ese tramo. Por ejemplo "Gire a la izquierda en la siguiente calle". También contendrá indicaciones de duración o distancia como: "Dentro de 5 kilómetros coja la salida, y continúe recto 40 minutos".

El objeto DirectionsStep consta de los siguientes campos:

- instructions: contiene instrucciones para este tramo en cadena de texto.
- distance: contiene la distancia que se recorre en este tramo.
- duration: contiene la duración que se recorre en este tramo.
- start address: contiene la latitud/longitud del origen de este tramo.
- end address: contiene la latitud/longitud del destino de este tramo.

5.2. Módulo Base de Datos

En esta sección detallaremos el módulo de la base de datos, así como el diagrama de nuestra base de datos.

5.2.1. MySQL

MYSQL [17] es un sistema de gestión de bases de datos relacional, multihilo y multi-usuario. MySQL AB, que desde Enero 2008 es una subsidiaria de Sun Microsystems y ésta a su vez de Oracle desde Abril 2009, desarrolla MySQL como software libre en un esquema de licenciamiento dual, donde los usos privativos deben sufragar el coste de una licencia, mientras que para el resto de usos se ofrece bajo licencia GNU GPL. MySQL es uno de los

SGBD más utilizados en aplicaciones web debido a su gran velocidad para consultas y su fácil integración en plataformas LAMP: Linux + Apache + MySQL + PHP (o Perl/Python), así como con Windows o Mac.

- **Alternativas:**

5.2.2. Oracle Database

Oracle [18] es un sistema de gestión de base de datos objeto-relacional desarrollado por Oracle Corporation, cuya licencia es privativa. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando:

- Soporte de transacciones
- Estabilidad
- Escabilidad
- Soporte multiplataforma

5.2.3. Microsoft SQL Server

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional.

PostgreSQL

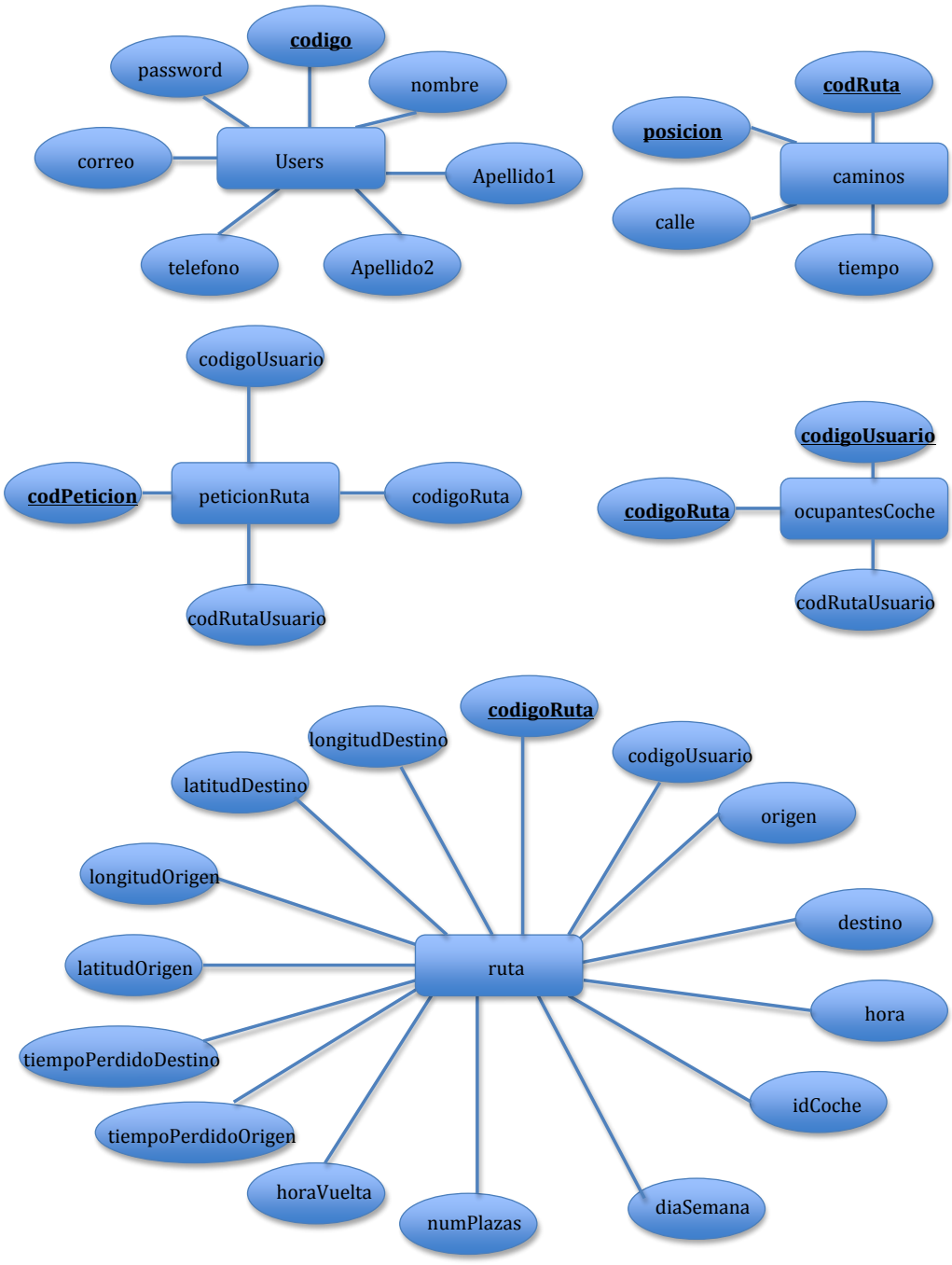
PostgreSQL es un sistema de gestión de base de datos relacional orienada a objetos y libre, publicado bajo la licencia BSD. La comunidad desarrolladora de PostgreSQL es la PGDG (PostgreSQL Global Developvemt Group).

- **Justificación de la elección de MySQL:** Elegimos MySQL por su sencillez, y oferta de características que cubren perfectamente las necesidades de este proyeto. Se opta por usar el motor InnoDB para las tablas de la base de datos para soportar

transacciones y mantener la integridad referencial. Este SGBD, para una distribución Linux que es la que usamos en el servidor, soporta unas 2000 conexiones simultáneas, lo cual es más que suficiente para el posible uso probable de la aplicación.

5.2.4. Diagrama de nuestra Base de Datos

En el siguiente diagrama se puede ver las tablas de las que consta nuestra base de datos:



- **Tabla Users:** En esta tabla tendremos la información personal de los usuarios de nuestro sistema. La clave primaria será un código generado de forma autoincremental.
- **Tabla ruta:** En esta tabla tendremos la información de todas las rutas del sistema, ya sean rutas con coche, o rutas sin coche, de un usuario, con los datos del origen y destino de la ruta, así como el día de la semana. La clave primaria será un código generado de forma autoincremental.
- **Tabla peticiónRuta:** Esta tabla la usaremos para guardar la información de las peticiones que hace un usuario en alguna ruta de las diversas que puede tener, para una ruta en concreto. La clave primaria será un código generado de forma autoincremental.
- **Tabla ocupantesCoche:** Esta tabla la usaremos para llevar un control de las rutas con coche que ya tienen asignados usuarios a los que recoger. En esta tabla asociaremos el código del usuario en una ruta suya, con el código de la ruta que le lleva en coche. Como un usuario no puede estar dos veces en una ruta con coche con varias paradas, nos bastará la tupla codigoRuta y codigoUsuario como clave primaria.
- **Tabla caminos:** En esta tabla guardaremos el orden en que se tienen que realizar las paradas dentro de una ruta con coche con varias paradas, así como el tiempo que se tarda entre ellas por motivos de eficiencia y la dirección de parada. Como clave primaria nos bastará la ruta, así como la posición que ocupa, ya que una ruta no puede pertenecer a más de una ruta con paradas.

5.3. Módulo Servidor Apache

En este módulo detallaremos el servidor Apache y las tecnologías relacionadas con él.

5.3.1. Servidor HTTP Apache

El servidor HTTP Apache [19] es un servidor web HTTP de código abierto, para plataformas Unix, Windows y Macintosh y otras, que implementa el protocolo HTTP y la

noción de sitio virtual. El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Apache tiene amplia aceptación en la red: desde 1996, Apache es el servidor HTTP más usado, alcanzando su máxima cuota de mercado en 2005 cuando el 70 % de los sitios web en el mundo usaban Apache. La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Las principales ventajas son:

- Modular
- Código abierto
- Multi-plataforma
- Extensible
- Popular (fácil conseguir apoyo o soporte)

Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web. Apache como ya hemos dicho anteriormente, es el componente de servidor web de la plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP/Perl/Python/Ruby. Este servidor web es redistribuido como parte de varios paquetes propietarios de software, incluyendo la base de datos Oracle y el IBM WebSphere application server. Mac OS X integra Apache como parte de su propio servidor web y como soporte de su servidor de aplicaciones WebObjects. También viene con muchas distribuciones linux. Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable.

Algunos de los sitios web más grandes del mundo están ejecutándose sobre Apache. Por ejemplo, la capa frontal del motor de búsqueda de Google está basado en una versión modificada de Apache denominada Google Web Server (GWS).

El principal competidor de Apache es Microsoft Internet Information Services (IIS), así como Sun Java System Web Server de Sun, pero nos hemos decantado por Apache dado el servidor que los tutores nos han facilitado.

5.3.2. HTTP (Hypertext Transfer Protocol)

HTTP es el protocolo usado en cada transacción de la web, que se ubica en la capa de aplicación de la pila de protocolos de internet. HTTP fue desarrollado por la World Wide Web Consortium (conocido por sus siglas W3C) y la Internet Engineering Task Force (IETF). En 1999, aparecieron publicados una serie de RFC fruto de la colaboración de ambas organizaciones, el más importante de ellos el RFC 216 que especifica la versión 1.1.

HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web(clientes, servidores, proveedores) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (un navegador web por ejemplo) se le conoce como user agent. A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos o URL. Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc. Además, HTTP es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Es por esto que se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de "sesión", la cual usamos en nuestro proyecto para guardar variables.

- **Justificación de su elección:** supone un estándar básico para la transmisión de ficheros en Internet y no existe alternativa mejor disponible para este proyecto.

5.4. Módulo Motor PHP

5.4.1. PHP (Hypertext Preprocessor)

PHP [9] es un acrónimo recurrente que significa PHP Hypertext Preprocessor. PHP es un lenguaje de programación interpretado de propósito general aunque está diseñado especialmente para el desarrollo web y puede ser empotrado en código HTML. Por lo general, se ejecuta en un servidor web, teniendo como entrada un código PHP y procesándolo, a la salida se obtiene un contenido web. Actualmente también se puede utilizar desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o QTK+. PHP apareció en 1995, influido por otros lenguajes como C, C++, Java, Perl y Python y resulta en un lenguaje multiparadigma, aunque con clara tendencia a ser un lenguaje imperativo y orientado a objetos. Desde el año 2000 este lenguaje empezó a despegar y se popularizó rápidamente por sus características parecidas a otros lenguajes de programación estructurada más comunes como C y Perl y que permiten a los programadores construir aplicaciones complejas con una curva de aprendizaje relativamente corta. [8]

- **Alternativas:**

  JSP/Java

  JavaServer Pages (JSP) [10] es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. Esta tecnología fue desarrollada por Sun Microsystems. Las JSP's permiten la utilización de código Java mediante scripts. JSP se puede considerar como una manera alternativa y simplificada de construir servlets. Es por ello que una página JSP puede hacer todo lo que un servlet puede hacer, y viceversa. Cada versión de la especificación de JSP está fuertemente vinculada a una versión en particular de la especificación de servlets.

  El funcionamiento general de la tecnología JSP es que el Servidor de Aplicaciones

interpreta el código contenido en la página JSP para construir el código Java del servlet a generar. Este servlet será el que genere el documento (tipicamente HTML) que se presentará en la pantalla del navegador del usuario.

CGI/Perl

CGI o lo que es lo mismo, Interfaz de entrada común, es una importante tecnología de la web, que permite a un cliente (navegador web), solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa cuyo resultado final de la ejecución son objetos MIME (Multipurpose Internet Mail Extensions). Las aplicaciones que se ejecutan en el servidor reciben el nombre de CGIs.

Las aplicaciones CGI fueron una de las primeras prácticas de crear contenido dinámico para las páginas web, de tal manera que esta interfaz rápidamente se volvió un estándar. En una aplicación CGI, el servidor web pasa las solicitudes del cliente a un programa externo. Este programa puede estar escritor en cualquier lenguaje que soporte el servidor, aunque por razones de portabilidad se suelen usar lenguajes de script. La salida de dicho programa es enviada al cliente en lugar del archivo estático tradicional.

Entre los lenguajes más habituales para escribir un programa CGI son: C, C++, Perl, Java, Visual Basic, etc. No obstante, debido a que el CGI recibe los parámetros en forma de texto será útil un lenguaje que permita realizar manipulaciones de las cadenas de caracteres de una forma sencilla, como por ejemplo Perl. Perl es un lenguaje interpretado que permite manipulaciones sencillas de ficheros y textos, así como la extracción y manipulación de cadenas de caracteres, unidas a unas búsquedas rápidas y fáciles.

Forma de actuar de un CGI: [11]

1. En primera instancia, el servidor recibe una petición (el cliente ha activado una URL que contiene el CGI), y comprueba si se trata de una invocación de un CGI.
2. Posteriormente, el servidor prepara el entorno para ejecutar la aplicación. Esta información procede mayoritariamente del cliente.
3. Seguidamente, el servidor ejecuta la aplicación, capturando su salida estándar.
4. A continuación, la aplicación realiza su función: como consecuencia de su actividad se va generando un objeto MIME que la aplicación escribe en su salida estándar.
5. Finalmente, cuando la aplicación finaliza, el servidor envía la información producida, junto con información propia, al cliente, que se encontraba en estado de espera.

Ruby on Rails

Ruby on Rails [12] es un framework de aplicaciones web de código abierto, escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador. Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real reescribiendo menos código que con otros frameworks y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible.

ASP.NET

ASP.NET [13] es la tecnología sucesora de ASP(Active Server Pages). ASP.NET es un framework para aplicaciones web, desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web diná micos, aplicaciones web y servicios web XML. ASP.NET está contruido sobre el Common Language Runtime,

permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

Las páginas de ASP.NET, conocidas oficialmente como formularios web (web forms), son el principal medio de construcción para el desarrollo de aplicaciones web. Los formularios web están contenidos en archivos con una extensión ASPX; en jerga de programación, estos archivos típicamente contienen etiquetas HTML o XHTML estático, y también etiquetas definiendo Controles Web que se procesan al lado del servidor, y Controles de Usuario, donde los desarrolladores colocan todo el código estático y dinámico requerido por la página web.

ASP.NET sólo funciona sobre el servidor de Microsoft IIS(Internet Information Services), lo que supone una desventaja respeto a otros lenguajes del lado del servidor, ejecutables sobre otros servidores más populares como Apache.

- **Justificación de la elección del PHP:** El problema común de las tecnologías anteriores, es que requieren de un entorno relativamente más complejo para su puesta en marcha. Además, PHP ofrece la posibilidad de ser desplegado en la inmensa mayoría de servidores web y casi todos los sistemas operativos y plataformas sin ningún coste económico. PHP ofrece un buen rendimiento y grado de desarrollo de numerosas funcionalidad disponibles (destacables por su robustez y sencillez), además de disponer de una gran comunidad de desarrolladores y documentación.

  PHP también ofrece una excelente conectividad con los principales SGBD(Sistemas Gestores de Bases de Datos), y una buena orientación a la programación orientada a objetos. Además, PHP ofrece una excelente integración con la plataforma LAMP (Linux + Apache + MySQL + PHP), que ofrece una plataforma de código abierto con unas características inmejorables para este proyecto en cuanto a nivel de prestaciones, rendimiento y facilidad de implantación.

  Por otra parte, el desarrollo en PHP es más ágil que el resto de lenguajes de este tipo,

incrementando la productividad en desarrollos de aplicaciones web que nos permiten centrarnos en aquello que se quiere conseguir y no en cómo se ha de codificar.

5.5. Módulo del Usuario

En esta sección detallaremos los aspectos relevantes en el ámbito del cliente(navegador).

5.5.1. JavaScript

Javascript [14] es un lenguaje de programación interpretado, basado en objetos, de tipado débil, liviano y se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje Java. Sin embargo, Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes. JavaScript es usado principalmente en páginas web, interpretado en el lado del cliente de la aplicación para conseguir mejores experiencias de uso de las interfaces. JavaScript proporciona un mecanismo de herencia basado en el paradigma de programación basada en prototipos, es decir, creando nuevas clases a partir de clonar una existente (base o prototipo) y extender su funcionalidad. La totalidad de navegadores modernos incorporan un analizador sintáctico o parser que les permite integrar JavaScript para poder interactuar con las páginas web, mediante una implementación del DOM. JavaScript a pesar de ser el más popular, es una implementación del lenguaje EMACScript.

- **Alternativas:**

  Adobe Flash

  Adobe Flash [15] tiene soporte para un lenguaje de programación interpretado conocido como ActionScript(AS) basado en el estándar ECMAScript. Desde su origen, ActionScript ha pasado de ser un lenguaje muy básico a un lenguaje avanzado con soporte de programación orientada a objetos, comparable en funciones y uso del lenguaje JavaScript.

Originalment creado para mostrar animaciones vectoriales en 2 dimensiones, ha pasado a convertirse en la opción preferida a la hora de crear aplicaciones Web que incluyen flujo de audio y video e interactividad. La utilización de gráficos vectoriales le permite disminuir el ancho de banda necesario para la transmisión, y, por ende, el tiempo de carga de la aplicación. Actualmente Flash Player está disponible para las versiones más recientes de los navegadores más porpulares(Internet Explorer, Mozilla Firefox, Safari, Opera). Google Chrome no lo necesita porque viene incluido dentro de él.

Microsoft Silverlight

Silverlight [16] es una estructura para aplicaciones web que agrega nuevas funciones multimedia como la reproducción de vídeos, gráficos vectoriales, animaciones e interactividad, en formar similar al ya mencionado Adobe Flash. Actualmente va por la versión 5.0 y se distribuye de forma gratuita. Además, se lanzó una versión en conjunto con Novell de Silverlight llamada Moonlight, la cual es código abierto para los sistemas operativos basados en Unix.

- **Justificación de la elección de JavaScript:** El uso de las tecnologías anteriores supone la obligación al usuario de tener instalador un complemento no estándar y de código no libre. Hemos elegido el uso de JavaScript debido a su amplio reconocimiento y porque supone la mejor opción para dotar la interfaz de una buena experiencia de usuario sin tener que recurrir a elementos externos por parte del usuario. Además, el uso de JavaScript vendrá motivado por el uso de Google Maps.

5.5.2. HTML (Hypertext Markup Language)

El lenguaje de marcado HTML tiene sus orígenes en el metalenguaje SGML (Standard Generalized Markup Language) diseñado para estructurar textos y relacionarlos en forma de hipertexto. La primera versión de este lenguaje definido por el SGML, apareció en 1991 de la mano de Tim Berners-Lee en un documento llamado HTML Tags. Este lenguaje es el

utilizado para describir la estructura y el contenido en forma de texto, mediante etiquetas, rodeadas por corchetes (<de apertura y >de cierre de etiqueta). Además de estructurar textos, puede incorporar otros elementos para describir su apariencia de forma limitada, así como imágenes y otros elementos que modifiquen la apariencia como son scripts y hojas de estilo, que veremos más adelante.

- **Justificación de su elección:** Elemento básico en cualquier página web.

5.5.3. CSS (Hojas de estilo en cascada)

El lenguaje de las hojas de estilo en cascada o CSS, nos permite definir la presentación de un documento estructurado escrito en HTML o XHTML. Este lenguaje, promovido por el W3C nace de la idea de separar la estructura de los documentos de su presentación, que queda en el ámbito de CSS. De esta manera, la información de estilo puede ser adjuntada como un archivo diferente o empotrado en el propio documento HTML/XHTML.

- **Justificación de su elección:** Es una buena práctica para poder separar contenido de presentación, toda la aplicación hará uso de CSS para su presentación, debido al ahorro de tamaño de las páginas, la claridad del código y la mantenibilidad (un cambio en CSS se refleja en la presentación de toda la aplicación).

5.5.4. Estándares de la web

Los estándares de la web, son una serie de tecnologías o soluciones aceptadas formal o globalmente debido a su correcta especificación y seguimiento de criterios de comportamiento regulado y estricto. Dentro de este conjunto, podemos encontrar los protocolos oficiales de Internet, recogidos en el RFC 5000, con categoría de memo. Para este apartado de estándares web, no se contemplan alternativas de uso posibles para este proyecto, más allá de las diferentes versiones o revisiones de los diferentes organismos responsables de los estándares.

5.6. Módulo del Servidor de Correo

5.6.1. Servidor de correo: Exim 4

Exim [22] (EXperimental Internet Mailer) es un agente de transporte de correo (Mail Transport Agent, usualmente MTA) desarrollado por la Universidad de Cambridge y puede ser utilizado en la mayoría de los sistemas Unix (entre ellos GNU/Linux). Si bien puede compilarse en sistemas operativos Windows, se recomienda que sea utilizado en producción sobre sistemas operativos de la familia Unix. Se distribuye sin costo bajo la licencia GNU GPL por lo que es, además, software libre. Tiene una gran flexibilidad en los caminos que pueden seguir los mensajes según su origen y por presentar funcionalidades para control de spam, listas de bloqueo basados en DNS (DNSBL), virus, control de relay, usuarios y dominios virtuales y otros, que se configuran y mantienen en forma más o menos sencilla. El proyecto cuenta con buena documentación, ejemplos y recetas claras de "como hacer" determinadas tareas. En términos generales se destaca que no existen situaciones para las que Exim sea una opción incorrecta y en muchas situaciones se desempeña como la mejor opción. Exim es el MTA por defecto en las distribuciones Debian GNU/Linux.

- **Alternativas:**

  Postfix

  Postfix [23] es un servidor de correo de software libre, un programa informático para el enrutamiento y envío de correo electrónico, creado con la intención de que sea una alternativa más rápida, fácil de administrar y segura al ampliamente utilizado Sendmail. Postfix es el agente de transporte por omisión en diversas distribuciones de Linux y en las últimas versiones del Mac OS X.

Sendmail

Sendmail es un popular agente de transporte de correo en Internet, cuya tarea consiste en encaminar los mensajes correos de forma que estos lleguen a su destino. Se afirma que es el más popular MTA, compatible con sistemas Unix y el responsable de la mayoría de envío del correo de internet, aunque se le critica su alto número de alertas de seguridad (la mayoría de ellas parcheadas a las pocas horas), además de no ser sencillo de configurar.

- **Justificación de la elección de Exim:** Sendmail la descartamos rápidamente, dado que como ya hemos dicho, es más difícil de configurar y tiene una larga historia de fallos. Entre Exim y Postfix realmente no estábamos seguros por que decantarnos, así que nos decantamos por Exim que era el que nos venía por defecto en el servidor, pero bien podríamos haber puesto postfix.

5.7. Módulo del administrador

En esta sección se detallaran las tecnologías que que usaremos como adminitradores para la interacción contra el servidor.

5.7.1. SSH (Secure Shell)

SSH es el nombre de un protocolo y del programa que lo implementa, que sirve para poder acceder a máquinas remotas a través de una red de forma segura. SSH trabaja de forma similar a como se hace con telnet, con la salvedad de que SSH usa técnicas de cifrado que hacen que la información que viaja por el medio de comunicación vaya de manera no legible y ninguna tercera persona pueda descubrir el usuario y contraseña de la conexción ni lo que se escribe durante toda la sesión.

- **Justificación de su elección:** Lo empleamos para conseguir una conexión segura con el servidor web, en detrimento de otras posibilidades como puede ser telnet.

5.7.2. SCP (Secure Copy)

SCP es un medio de transferencia de archivos entre un host local y otro remoto, o entre dos hosts remotos, usando el protocolo SSH.

- **Justificación de su elección:** Lo empleamos para transferir archivos de forma segura desde nuestro ordenador al servidor web si no estamos en el entorno de dreamweaver.

5.7.3. SFTP (SSH File Transfer Protocol)

SFTP es un protocolo del nivel de aplicación que proporciona la funcionalidad necesaria para la transferencia y manipulación de archivos sobre un flujo de datos fiable. Se utiliza comúnmente con SSH para proporcionar la seguridad a los datos, aunque permite ser usado con otros protocolos de seguridad. Por tanto, la seguridad no la provee directamente el protocolo SFTP, sino SSH en este caso. A diferencia de SCP, SFTP permite más operaciones que SCP, el cual solo permite la transferencia o copia de archivos. SFTP intenta ser más independiente de la plataforma que SCP.

- **Justificación de su elección:** Lo empleamos para subir todo el espacio de trabajo o los cambios realizados, desde el entorno de Dreamweaver. Usamos SFTP y no otros como FTP por la seguridad que implementa.

5.7.4. SVN

SVN es un sistema de control de versiones usado para que varios desarrolladores puedan trabajar en un mismo proyecto en forma más o menos ordenada. Tiene una arquitectura cliente servidor con controles de concurrencia para cuando varios desarrolladores están trabajando en el mismo archivo y funciona más o menos así. En algún servidor se monta un repositorio SVN. En este lugar se van a registrar los cambios (revisiones) y los logs que se vayan generando. El cliente de SVN se baja una copia local de alguna revisión (generalmente la última), el desarrollador hace los cambios y los sube al servidor para que estén disponibles

para los otros desarrolladores (además de generar un log con un comentario de que cosas modificó).

Subversión es un sistema de control de versiones diseñado específicamente para reemplazar al popular CVS. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como SVN por ser el nombre de la herramienta utilizada por línea de comandos.

Todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en un instante determinado del repositorio que se está trabajando.

Subversión puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintas computadoras. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada. SI hay problemas, siempre se pueden deshacer los cambios y volver a un punto anterior.

Son varios los clientes que se pueden usar, entre los que destacamos: TortoiseSVN, Subclipse (integrado en Eclipse), ViewVC y RapidSVN. Sin embargo, nosotros haremos uso del SVN por línea de comandos, para lo cual, explicaremos los comandos más usados: [23]

- svn checkout svn://source.dacya.ucm.es/forja/scmrepos/svn/carpool : Para obtener una copia del trabajo en local tenemos que conocer la ubicación de nuestro repositorio en internet. Con el comando checkout se nos descargará el contenido del repositorio en nuestro ordenador.

- svn help : Nos mostrará una lista con todos los comandos de svn disponibles.

- svn status: Así podemos conocer el estado de la copia de trabajo.

- svn add archivo: Para agregar un archivo o una careta y todo su contenido al control de versiones.
- svn del archivo: Para eliminar un archivo o una carpeta y todo su contenido al control de versiones.
- svn update : Cada vez que volvemos a trabajar sobre una copia de un repositorio pasado cierto tiempo, es necesario asegurarnos de que estamos trabajando sobre la última copia del repositorio.
- svn log : Para saber que usuario modificó cierto archivo y que cambios realizó.
- svn diff archivo: Para ver los cambios realizados a un archivo en particular entre la versión actual y la anterior.
- svn cleanup : Para solucionar posibles conflictos.
- svn commit: Para realizar el commit y guardas los cambios realizados en el repositorio.

Cabe destacar que para poder trabajar con el svn por línea de comandos, será necesario darle valor a la variable de entorno SVN_EDITOR, para el cual usaremos el editor vi, aunque se podría usar otros como emacs. En concreto: export SVN_EDITOR=vi;

5.7.5. Sistema de documentación LaTeX

LaTeX [2] es un sistema de composición de textos que está formado mayoritariamente por órdenes (macros) construidas a partir de comandos de TeX(un lenguaje "de bajo nivel", en el sentido de que sus acciones son muy elementales). LaTeX fue escrito por Leslie Lamport en 1984, mientras que TeXfue creado por Donald knuth. Está orientado especialmente a la creación de libros, documentos cientfícos y técnicos que contengan fórmulas matemáticas, así como para presentaciones de proyectos de fin de carrera.

Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos, dado que la calidad tipográfica de los documentos realizados con LaTeX es comparable a la de una editorial científica de primera línea.

LaTeX es *software* libre bajo licencia LPPL (Licencia Púbica del Proyecto LaTeX) y presupone una filosofía de trabajo diferente a la de los procesadores de texto habituales basándose en comandos. Tradicionalmente, este aspecto se ha considerado una desventaja, sin embargo, LaTeX, a diferencia de los procesadores de texto comunes, permite a quien escribe un documento centrarse exclusivamente en el contenido sin tener que preocuparse de los detalles del formato. Además de sus capacidades gráficas para representar ecuaciones, fórmulas, notación científica e incluso musical, permite estructurar fácilmente el documento (con capítulos, secciones, notas, bibliografía, índices analíticos, etc.) lo cual brinda comodidad y lo hace útil para artículos académicos y libros técnicos. Una de las ventajas de LaTeX es que la salida que ofrece es siempre la misma, con independencia del dispositivo (impresora, pantalla, etc.) o el sistema operativo (MS Windows, MacOS, Unix, GNU/Linux, etc.) y puede ser exportado a partir de una misma fuente a numerosos formatos tales como Postscript, PDF, SGML, HTML, RTF, etc. [3]

5.7.6. Programas empleados

A continuación detallamos los programas que hemos utilizado para el desarrollo del proyecto, todos ellos para el sistema operativo Mac OS 10.7 Lion, que es con el que hemos trabajado los 3 integrantes del proyecto, salvo el Microsoft Visio 2010 que tiene que ser desde un Windows.

Adobe Dreamweaver CS5.5

Adobe Dreamweaver [24] es una aplicación en forma de suite, que está destinada a la construcción, diseño y edición de sitios y aplicaciones Web basados en estándares. Creado inicialmente por Macromedia (actualmente producido por ADobe Systems), es el programa más utilizado en el sector del diseño y la programación web, por sus funcionalidad, su

integración con otras herramientas como Flash, y por su soporte reciente a los estándares del World Wide Web Consortium. Dreamweaver ha tenido un gran éxito desde finales de los años 1990, y actualmente mantiene el 90 % del mercado de editores HTML. Esta disponible de forma nativa tanto para MAC como para Windows.

XAMPP

Para trabajar en local e ir haciendo pruebas y ver como iba quedando el proyecto, hemos usado XAMPP [25]. XAMPP es un servidor independiente de la plataforma, de software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas diná micas. Actualmente XAMPP está disponible para Windows, GNU/Linux, Solaris y Mac OS X.

MySQL Workbench

MySQL Workbench [26] es la herramienta que hemos utilizado para el manejo de nuestra base de datos MySQL, la cual integra desarrollo de software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.

TeXShop

Es el programa que hemos usado para el desarrollo de esta memoria en LaTeX. TeXShop [27] es una interfaz gráfica para la visualización y edición del código LaTeX puesto que dispone de opciones muy interesantes tales como los códigos de colores para la sintaxis, un correcto ortográfico y la construcción de macros, todo ello estructurado en torno al motor de código LaTeX para la compilación. Además, dispone de la ventaja que te avisa de la línea si hay algún fallo de compilación.

Microsoft Visio 2010

Para la realización de los diagramas UML, hemos utilizado el Microsoft Visio 2010. Las herramientas que lo componen permiten realizar diagramas de oficinas, diagramas de bases de datos, diagramas de flujo de programas, UML, y más, que permiten iniciar al usuario en los lenguajes de programación.

Adobe Illustrator

Para el diseño de nuestro logo, así como algún retoque más, hemos utilizado el Adobe Ilustrator. Se trata esencialmente de una aplicación de creación y manipulación vectorial en forma de taller de arte que trabaja sobre un tablero de dibujo, conocido como mesa de trabajo y está destinado a la creación artística de dibujo y pintura para Ilustración (Ilustración como rama del Arte digital aplicado a la Ilustración técnica o el diseño gráfico, entre otros).

Capítulo 6

Algoritmos para la generación y consulta de rutas

En esta sección describiremos los algoritmos más importantes que hemos desarrollado para el proyecto.

6.1. Algoritmo de cálculo de ruta

Para calcular la ruta óptima que debe seguir el coche en una ruta con varios usuarios, hacemos uso del algoritmo de cáculo de ruta de Google Maps que explicado en el módulo de Google Maps. En concreto, nos ayudaremos de la ruta óptima que se calcula como ya explicamos con el atributo optimizeWaypoints=True. Sin embargo, si usásemos sin más el algoritmo, usando de puntos intermedios los orígenes y destinos de los usuarios que tenemos que recoger, se podría dar el caso de que el coche pasara antes por el destino de un usuario antes que por su origen, cosa que no deseamos.

En una primera aproximación, se podría pensar en la fuerza bruta, y calcular todas los posibles ordenes de las posibles rutas, y cálculando para cada una de ellas el tiempo que se tarda en recorrerla usando el algoritmo del cáculo de ruta de Google Maps, lo cual no es muy aconsejable, ya que el número de solicitudes a Google sería enorme, solicitudes que al ser javascript se ejecutan en el cliente. Esto último hoy en día en servicios como el móvil, donde el usuario tiene una tarifa tope de datos, sería inadmisible.

Para evitarnos tener que hacer tantas peticiones, pero seguir calculando la ruta óptima, nos basaremos en la idea de que si calculásemos la ruta óptima únicamente con los puntos de origen como puntos intermedios, estos estarán en el mismo orden que los puntos de origen calculados junto con sus destinos con el algoritmo de cáculo de ruta de Google Maps.

De este modo, el algoritmo en pseudocódigo quedaría así:

```
1
2 origenRuta = Origen de la ruta con coche;
3 destinoRuta = Destino de la ruta con coche;
4
5 puntosIntermedios= Orígenes de los puntos intermedios;
6
7 while (puntosIntermedios != vacío) {
8
9   siguienteParada=calculoRutaOptima(origenRuta,destinoRuta,puntosIntermedios)
10
11    #Actualizamos la siguiente parada de la ruta
12    origenRuta=siguienteParada;
13
14    IF siguienteParada = origen de un usuario THEN quitamos el origen de los
          puntosIntermdios, y añadimos su destino a los intermedios.
15
16          ELSIF siguienteParada = destino de un usuario THEN  simplemente lo
              quitamos de puntosIntermedios.
17 }
```

En el peor de los casos, en una ruta con 4 ocupantes el coche, se ejecutará únicamente 7 veces.

6.2. Algoritmo de filtrado

A la hora de hacer peticiones para apuntarse a una ruta con coche, hemos hecho un algoritmo de filtrado para que al usuario no le salgan todas las rutas disponibles en el sistema. Básicamente, consta de 3 partes o filtrados:

6.2.1. Filtrado por hora y día

Al usuario solo le mostraremos rutas que encajen con la hora y día de la ruta con coche que encaje con su ruta de la que quiere pedir que le lleven. En concreto, como los usuarios tienen flexibilidad de horario en el origen y destino, se calculará la fecha de salida más pronto

posible de la ruta con coche, así como su fecha de llegada más tardía, y se comprobará que la hora de la ruta que quiere formar parte de ella, encaja.

6.2.2. Filtrado por proximidad

La aplicación maneja coordenadas (latitud y longitud) para ubicar a los orígenes y destinos de las rutas. Para el filtrado, usaremos un algoritmo que determina si un punto P está dentro de un polígono convexo. Dado un segmento [A,B] y un punto P, el producto vectorial puede utilizarse para saber si el punto está a la derecha o a la izquierda del segmento.

Sea $\overrightarrow{AB}$ el vector que forma el segmento [A,B]. Sea $\overrightarrow{ABp}$ el vector perpendicular a $\overrightarrow{AB}$.

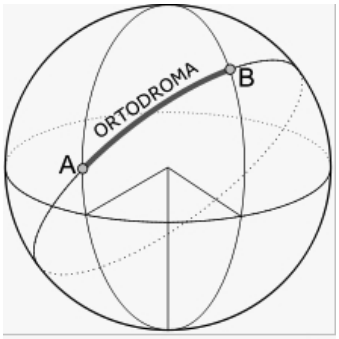
- Si $\overrightarrow{AP} * \overrightarrow{ABp} > 0$, entonces P está a la izquierda de $\overrightarrow{AB}$.
- Si $\overrightarrow{AP} * \overrightarrow{ABp} < 0$, entonces P está a la derecha de $\overrightarrow{AB}$.
- Si $\overrightarrow{AP} * \overrightarrow{ABp} = 0$, P está encima de la recta(perpendicular a la normal).

Utilizando esta propiedad, si el punto se encuentra a la izquierda de todos los segmentos que forman el polígono, entonces el punto es interior. Como el polígono se asumió que es convexo, basta que el punto esté a la derecha de un solo segmento para que se considere que está fuera del polígono. Se asume que las aristas se enumeran en sentido antihorario.

Aplicado a nuestro proyecto, calcularemos los puntos (latitud/longitud) que conforman el rectángulo que contiene a los puntos de origen y destino de las rutas con coche a las que nos podemos apuntar por hora, y comprobaremos que nuestro origen y nuestro destino está dentro de ese rectángulo o no. En caso de que esté, se le ofrecerá al usuario como posible ruta a pedir.

Para calcular el ancho del rectángulo, que es la mitad que el largo, hacemos uso de la fórmula del Haversine [31] para una mayor exactitud. Como ya hemos comentado, manejamos coordenadas (latitud y longitud) para ubicar los usuarios en el espacio y poder realizar los cálculos de cercanía entre unos y otros. Para saber la distancia entre 2 coordenadas, no nos sirve hacer el cáculo en el plano Euclidiano, a pesar de que el error no será excesivo.

Lo que hacemos es calcular la distancia Ortodrómica, que es la distancia más corta entre dos puntos de la superficie de una esfera, medida a lo largo de un camino sobre esta; es el arco del círculo máximo que los une, menor de 180 °.



Distancia ortodrómica entre dos puntos a lo largo de un círculo máximo sobre la superficie de una esfera.

Para este cálculo usamos la fórmula del Haversine, que nos da la distancia del círculo máximo entre dos puntos, a partir de sus latitudes y longitudes. Es un caso especial de una fórmula más general de trigonometría esférica, la ley de los semiversenos, que relaciona los lados y los ángulos de "triángulos esféricos".

Fórmula del Haversine:

$$\operatorname{haversin}\left(\frac{d}{R}\right) = \operatorname{haversin}(\varphi_1 - \varphi_2) + \cos(\varphi_1)\cos(\varphi_2)\,\operatorname{haversin}(\Delta\lambda).$$

donde:

- haversin es la función haversine, $haversin(\theta) = sen^2(\frac{\theta}{2}) = \frac{(1-cos(\theta))}{2}$
- d es la distancia entre los dos puntos (a lo largo de un círculo máximo de la esfera)
- R es el radio de la esfera
- ϕ 1 es la latitud del punto 1

- ϕ 2 es la latitud del punto 2
- $\Delta\lambda$ es la diferencia de longitud
- El argumento de la función haversine se entiende en radianes.

Para utilizar la Fórmula del Haversine necesitamos, además de las dos posiciones (lat + lon), el radio de la Tierra. La fórmula asume que la Tierra es completamente redonda, con lo que cabe esperar una tasa de error que se podría llegar a asumir.Este valor es relativo a la latitud, donde el valor del radio ecuatorial es de 6378 km mientras que el polar es de 6357 km. El radio equivolumen es de 6371 km.

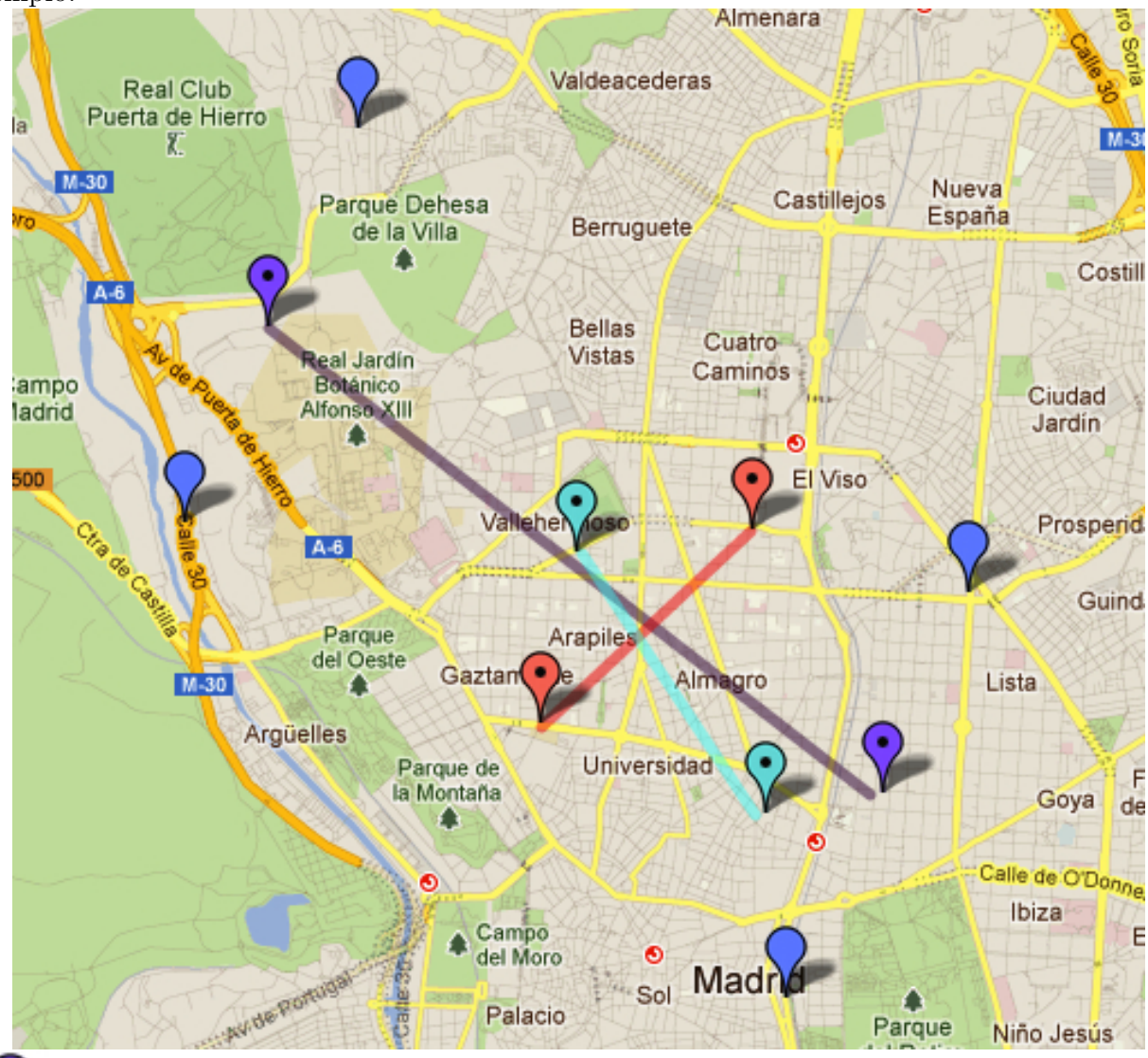
El código JavaScript para el cáculo de la distancia entre 2 puntos latitud/longitud es el siguiente:

```
1
2   function distanciahaverline(origen,destino)
3   {
4    var earth = 6371;
5
6   var lat1 = deg2rad(origen.lat);
7   var long1= deg2rad(origen.long);
8   var lat2 = deg2rad(destino.lat);
9   var long2= deg2rad(destino.long);
10
11  //Haversine Formula
12  var dlong=long2-long1;
13  var dlat=lat2-lat1;
14  var sinlat=sin(dlat/2);
15  var sinlong=sin(dlong/2);
16  var a=( sinlat*sinlat)+cos(lat1)*cos(lat2)*(sinlong*sinlong);
17  var c=2*asin(min(1,sqrt(a)));
18
19  var d=round(earth*c);
20
21  return d;
22  }
```

6.2.3. Filtrado por ángulo

Además de lo anterior, añadimos la condición de que el ángulo que forma el vector que une el origen y destino de la ruta con coche, con el ángulo que forma el vector que une el

origen y el destino de la ruta sin coche, deben de formar un ángulo de +- 90 °. Veamos un ejemplo:



Ruta 1 con coche: calle hermosilla 20 - calle Profesor García Santesmases (Facultad de Informática, Universidad Complutense de Madrid)

Puntos que conforman el rectángulo de nuestro algoritmo en base a la ruta 1

Ruta 2 sin coche: calle argensola 10 - avenida filipinas 22

Ruta 3 sin coche: calle Alberto Aguilera 40 - calle ríos rosas 52

A la hora de pedir para una ruta sin coche, al usuario se le mostrarán todas las rutas con coche, que pasen los filtrados anteriores. En este caso, para el usuario de la ruta 2 sin coche, se le ofrecerá como posibilidad de petición de ruta, la ruta 1, ya que tanto el origen como el destino de las rutas sin coche se encuentran dentro del rectángulo que forma la ruta sin coche, y a su vez, cumplen la restricción del ángulo de +- 90 º. En concreto, el vector que forma la ruta 1, con el vector que forma la ruta 2, se ve a simple vista que forma un ángulo de menos de 90 º, por tanto es apto.

Sin embargo, en el caso de la ruta 3, formaría un ángulo de 114,44 º, así que para esa ruta en esa dirección no se le mostraría la ruta 1 como opción posible para que le lleve en coche. Sin embargo, si la ruta 3 fuera a la inversa, es decir, de la calle Rio Rosas 52 a la calle Alberto Aguilera 40, si que se le mostraría la ruta 1 como posible ruta a pedir, ya que formaría un ángulo de 65,65 º.

6.3. Algoritmo de encaje de horas de una ruta

En las rutas, a la hora de hacer que todas encajen en la hora que desean, realizamos el siguiente algoritmo:

```
1
2 Hora= hora_inicio_ruta;
3 Para cada j en paradas(ruta) hacer
4     Si entra_en_tiempos(Hora,j)
5             Solucion.add(Hora)
6             Hora=Hora+tiempo_al_proximo_punto
7     sino
8             si es_menor(Hora,horquilla)//llega antes de la hora que quiere el
                  usuario
9                     diferencia=Hora-
                          hora_limite_mas_temprana_que_puso_el_suario
10                    Hora=hora_limite_mas_temprana_que_puso_el_suario
11                    Retrasa(Solucion, diferencia)
```

```
12              Sino
13                          diferencia= hora_limite_mas_tardia_que_puso_el_suario-
                                Hora
14                          Hora= hora_limite_mas_tardia_que_puso_el_suario
15                          Adelanta(Solucion, diferencia)
16              fsi
17              Solución=Hora
18          fsi
19  fin para
```

Capítulo 7

Conclusión:

7.1. Conclusiones

Como ya hemos comentado anteriormente el Carpooling ofrece muchas ventajas:

- Potenciar un uso más racional del coche
- Menor contaminación atmosférica y de ruido
- Mejor uso del espacio público
- Ahorro de combustible y compartir gastos
- Ahorro de tiempo

Aunque estas ventajas son claras, la mayoría de los españoles son bastantes escépticos en el uso de esta nueva manera de moverse. Esto es debido a una serie de inconvenientes:

- La posible impuntualidad
- Estar sujeto a un horario fijo
- Acordar los horarios de recogida y llegada
- Y sobre todo por el miedo, inseguridad de viajar con una persona desconocida

El mayor problema es la inseguridad ante un extraño, especialmente por la falta de responsabilidad de este, ya que puede darse el caso de que sea impuntual o incluso peor, no pasar a recoger a la persona. Esta idea viene impuesta por la sociedad de hoy en día en la que cada individuo piensa en su comodidad sin importarle la de los demás, sin renunciar a nada para que los individuos de alrededor se puedan sentir más cómodos.

Se debería inculcar o poner mas énfasis en valores como la responsabilidad, la empatía, el compañerismo... Pero esta misión no debe ser sólo de los colegios y educadores sino también se debe educar en las familias.

El sistema de Carpooling para viajes largos funciona bastante bien en Europa en general (en España cada vez más). El problema del Carpooling hoy en día reside en la utilización de este sistema en el día a día (como por ejemplo para ir al trabajo), que es en estos trayectos diarios cuando de verdad se notan las ventajas o beneficios antes mencionados.

Como solución al problema del uso del sistema Carpooling diariamente, se podría crear un sistemas de Carpooling cerrado, es decir en universidades, empresas, colegios... Esta solución crearía menos inseguridad ante la persona desconocida que viaje contigo, ya que saber que es de tu empresa,universidad... genera mayor confianza.

7.2. Portabilidad

En esta sección comentaremos brevemente la portabilidad a otras plataformas como puede ser el móvil, o el uso de esta aplicación en otros entornos.

7.2.1. Portabilidad a otras plataformas

Las empresas hoy en día necesitan que sus aplicaciones web se encuentren en internet y se puedan visualizar en diferentes dispositivos, y unos de los principales que está teniendo un gran auge son los dispositivos móviles, para lo cual el patrón de diseño Modelo Vista Controlador que implementamos, es esencial. Con esta arquitectura, los desarrolladores web tiene un mejor control de la aplicación y con ello, más facilidad en una futura portabilidad

a una plataforma móvil como Android o Mac os.

Pero implementar una aplicación para dispositivos móviles en la actualidad y que sea universal para que la aplicación funcione en cualquier tipo de dispositivo móvil, no es algo tan sencillo. El Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos como ya hemos explicado. Esto hace que sea más fácil la portabilidad a otras plataformas, ya que modificando la vista, y en menor medida el controlador, bastaría, sin tener que rediseñar el sitio por completo. [33]

7.2.2. Portabilidad a otros entornos

El carpooling está enfocado para viajes largos o trayectos cortos del día a día en este caso especialmente para trayectos al lugar de trabajo. Pero este sistema se puede llevar a otro entornos como:

- Transporte de mercancías:

  Haciendo unas leves modificaciones en el sistema, se podría utilizar la idea del Carpooling para crear rutas para transportistas, ya que son muchas veces las que tienen que transportar mercancías a otro lugares y cuando vuelven de estos vienen vacío. Utilizando un sistema de Carpooling específico para este entorno se podría conseguir que el transportista volviera con mercancía y así con seguir un mayor rendimiento, eficacia y ahorro tanto para los transportistas como para los proveedores.

- Transporte al aereopuerto:

  Este es otro entorno donde se podría sacar beneficios a la idea del Carpooling. Algunas empresas se dedican a traer o llevar a gente del aeropuerto al hotel o a su casa, o viceversa. Muchos de estos trayectos han sido contratados con anterioridad, por la tanto se podrían crear trayectos (con el sistema Carpooling) que vayan recogiendo a está gente y así compartir vehículo. Lo que significaría menor coste tanto para el usuario como para la empresa.

Apéndice A

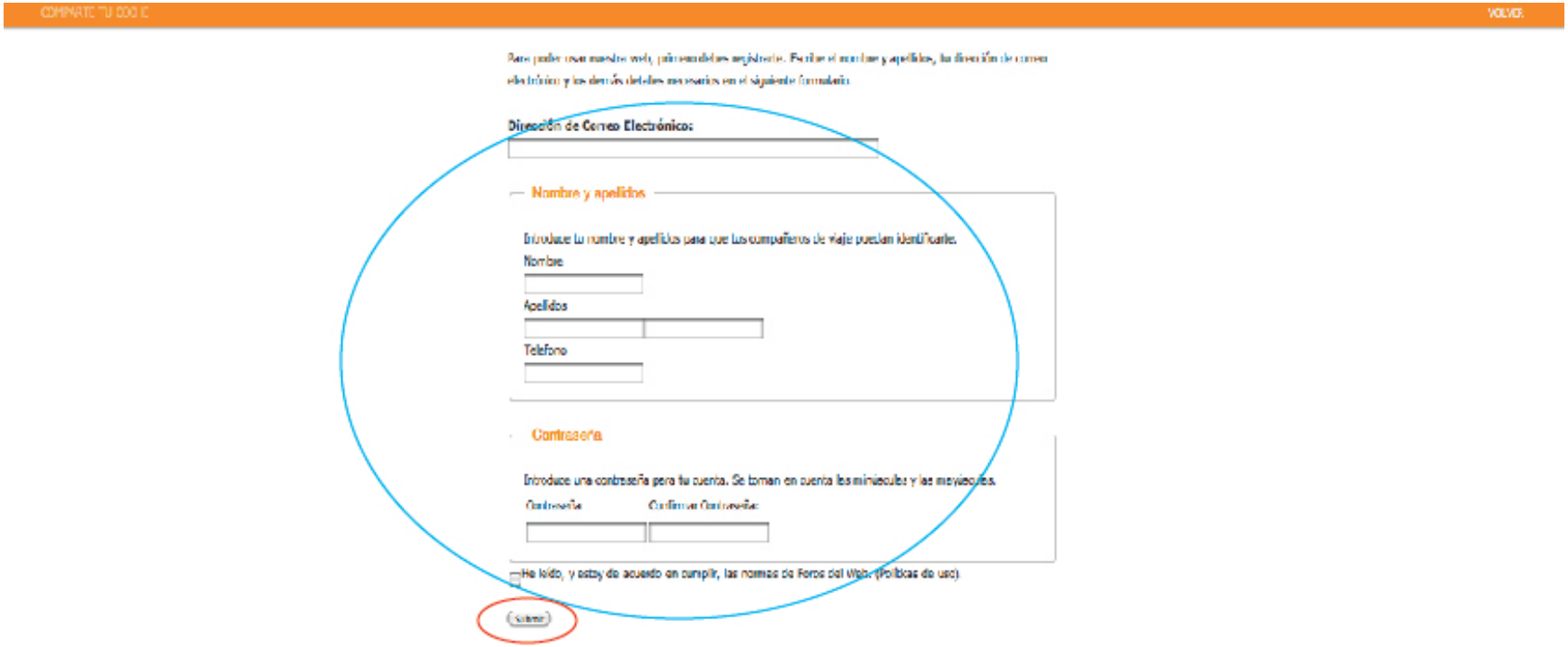
Como utilizar la aplicación web

A continuación se incluyen diversas capturas de pantalla que ilustran diversos casos de uso habituales de la aplicación desarrollada.

LOGIN

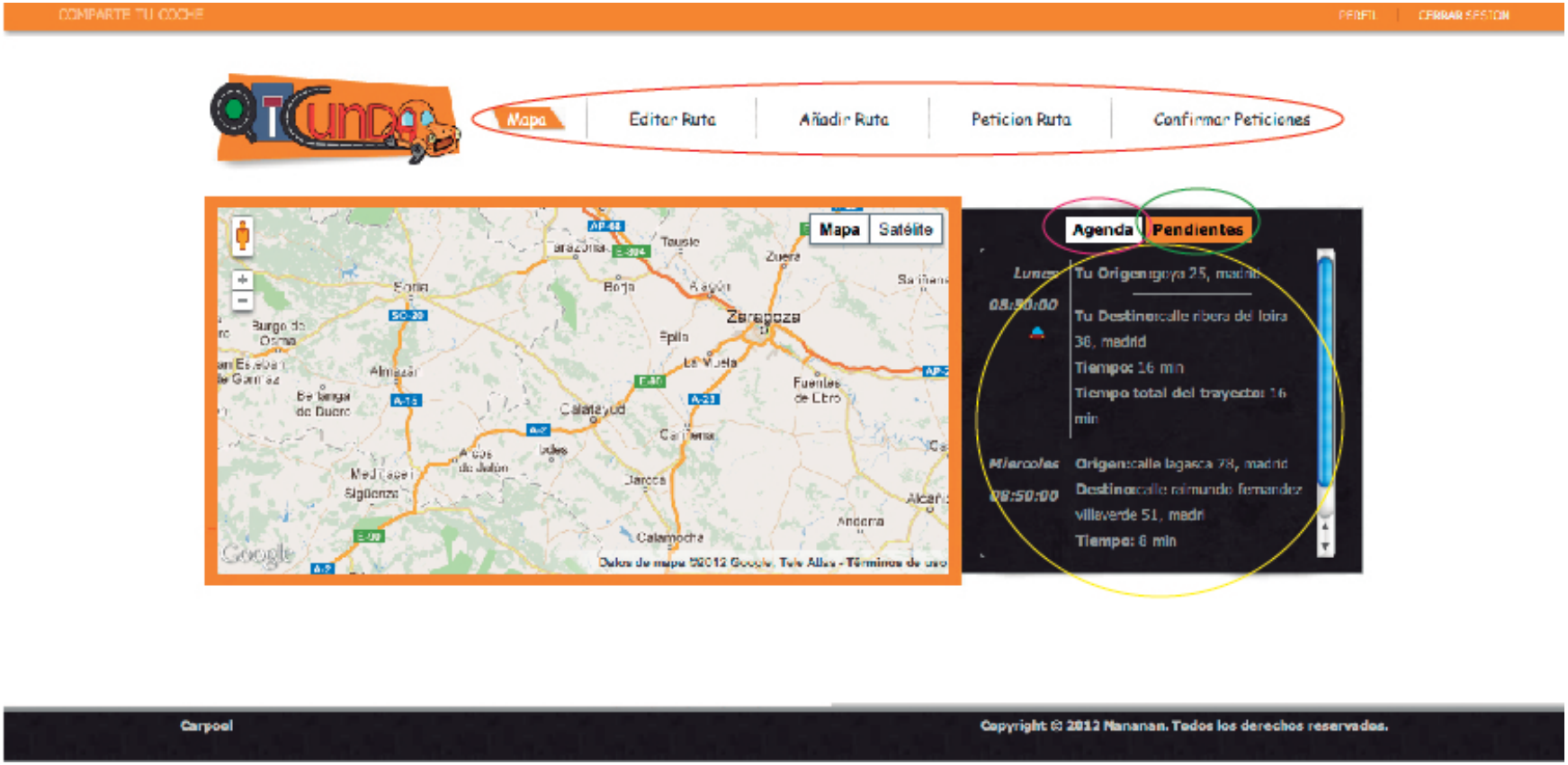
Introduce tu correo electrónico y password y pulsa Entrar, si aun no estás registrado pulsa en Registrarse.

REGISTRO



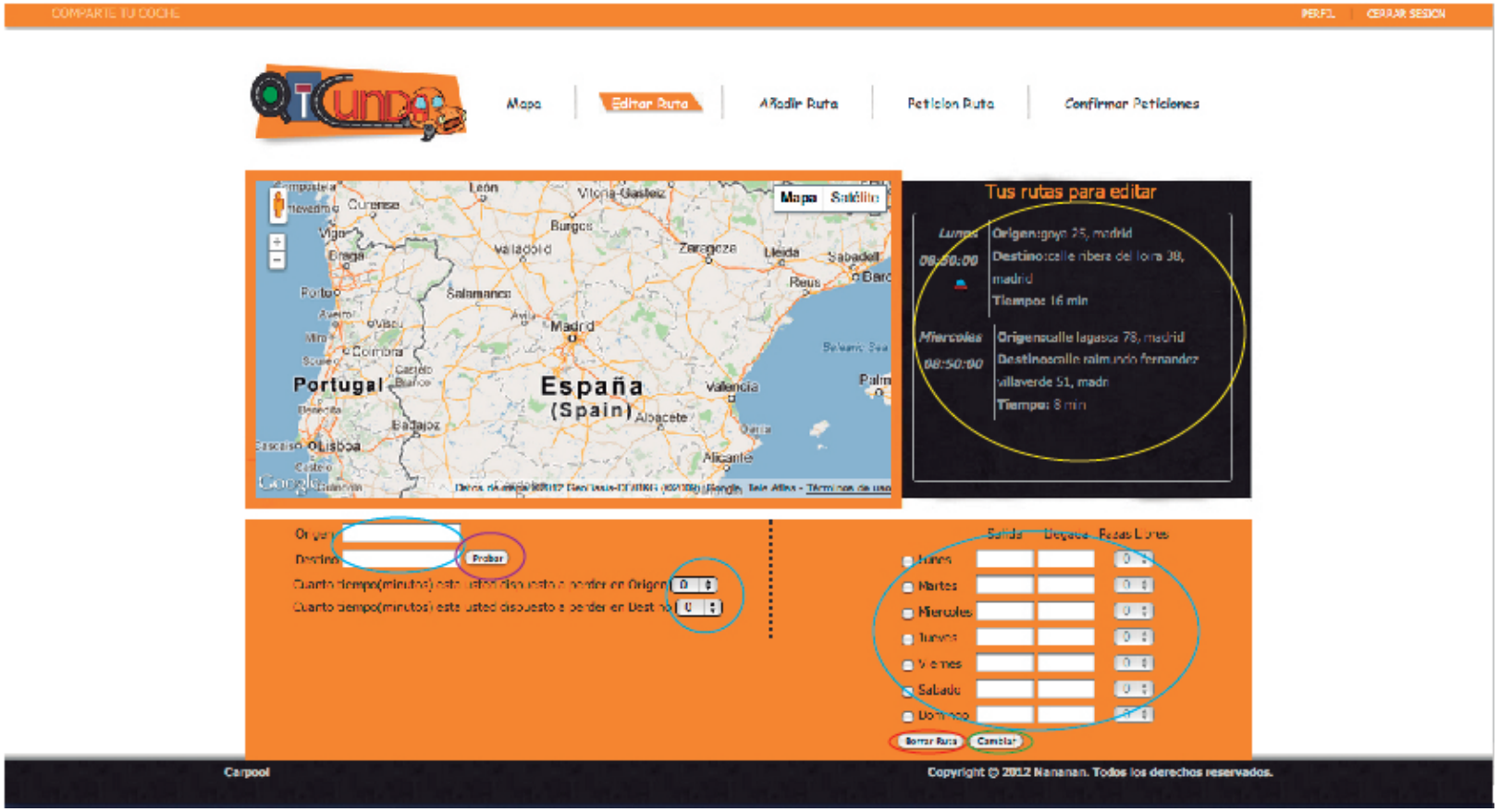
Rellene los diferentes Campos y pulse el Botón Registrar para confirmar el registro.

MAPA



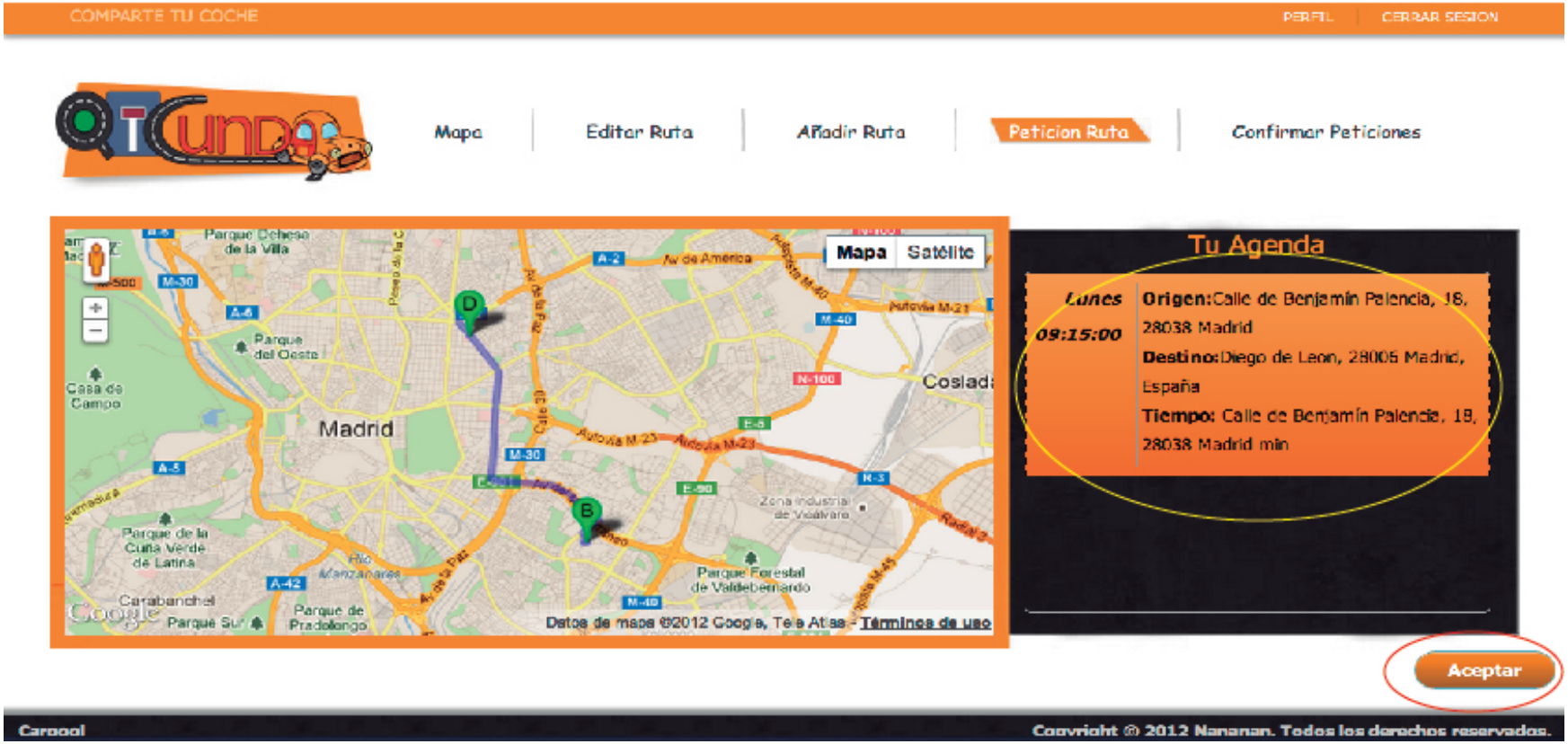
Pinche en el Cuadro de la derecha para ver cada una de las rutas en el mapa. Haga click en Agenda para ver sus rutas y si pincha en Pendientes nos muestra las rutas pendientes que tiene de ser aceptadas por otros usuarios. Puede desplazarse por el Menú de arriba para moverse entre las distintas opciones.

EDITAR RUTA



Pinche en el Cuadro para ver la ruta que quieres editar. Rellene o seleccione los diferentes Campos que quiere cambiar de la ruta. Puede borrar una ruta haciendo click en en el Botón Borrar Ruta y si quieres editarla pulsa en el Botón Cambiar. Para probar una ruta antes de cambiarla pulse en el Botón Probar.

PETICIÓN RUTA

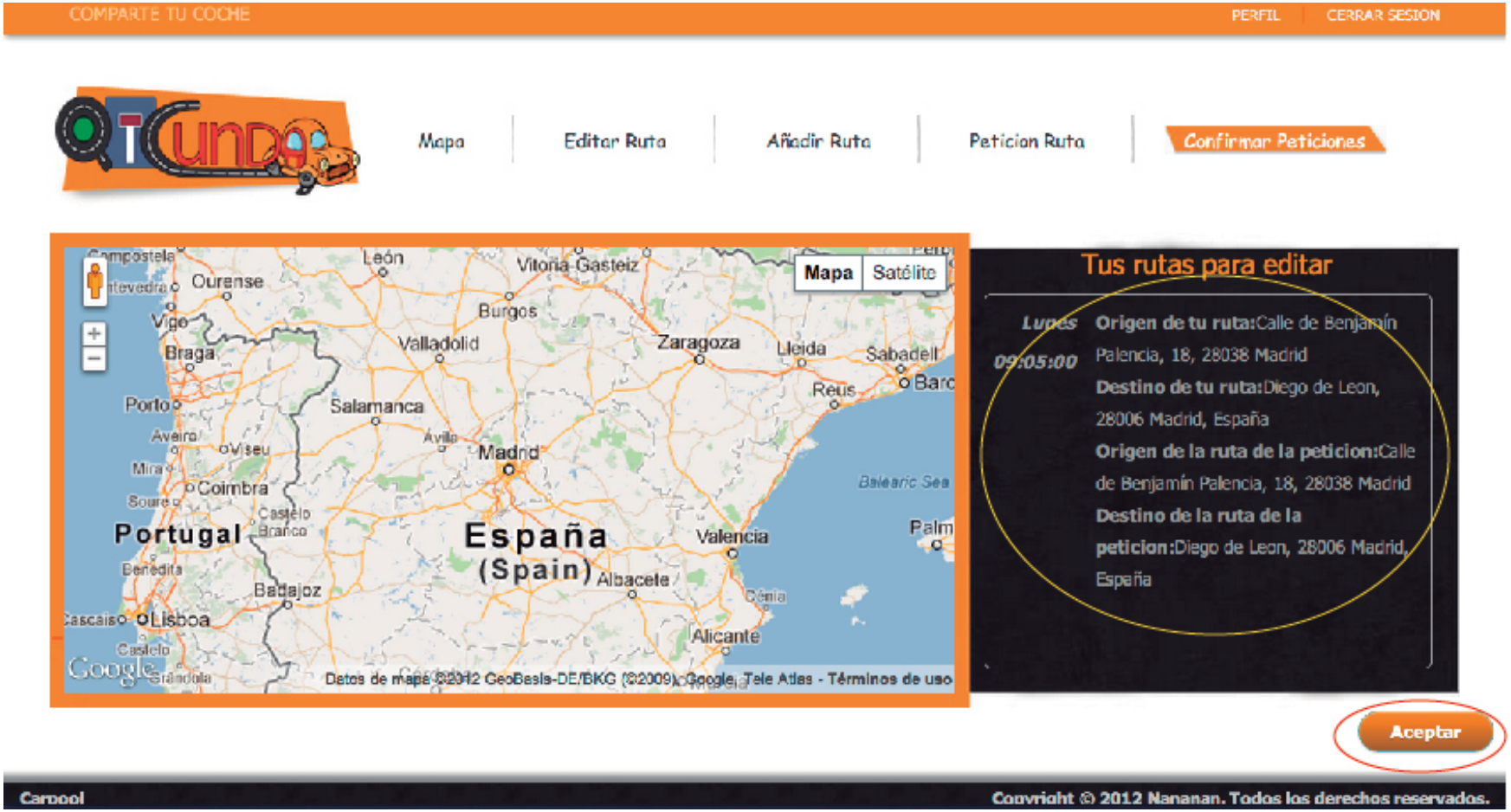


Seleccione la ruta del Cuadro de la que quieres enviar una petición, automáticamente te aparecerá en el Cuadro las diferentes peticiones que puedes hacer sobre tu ruta, selecciona la petición que desees y automáticamente se actualizará el mapa con la ruta provisional, si esta conforme pulse el Botón Aceptar para confirmar la petición.

AÑADIR RUTA

Rellene los diferentes Campos con las características que quiere que tenga su ruta. Haga click sobre el Botón Probar para ver la ruta antes de crearla. Pinche en el Botón Crear para crear la ruta.

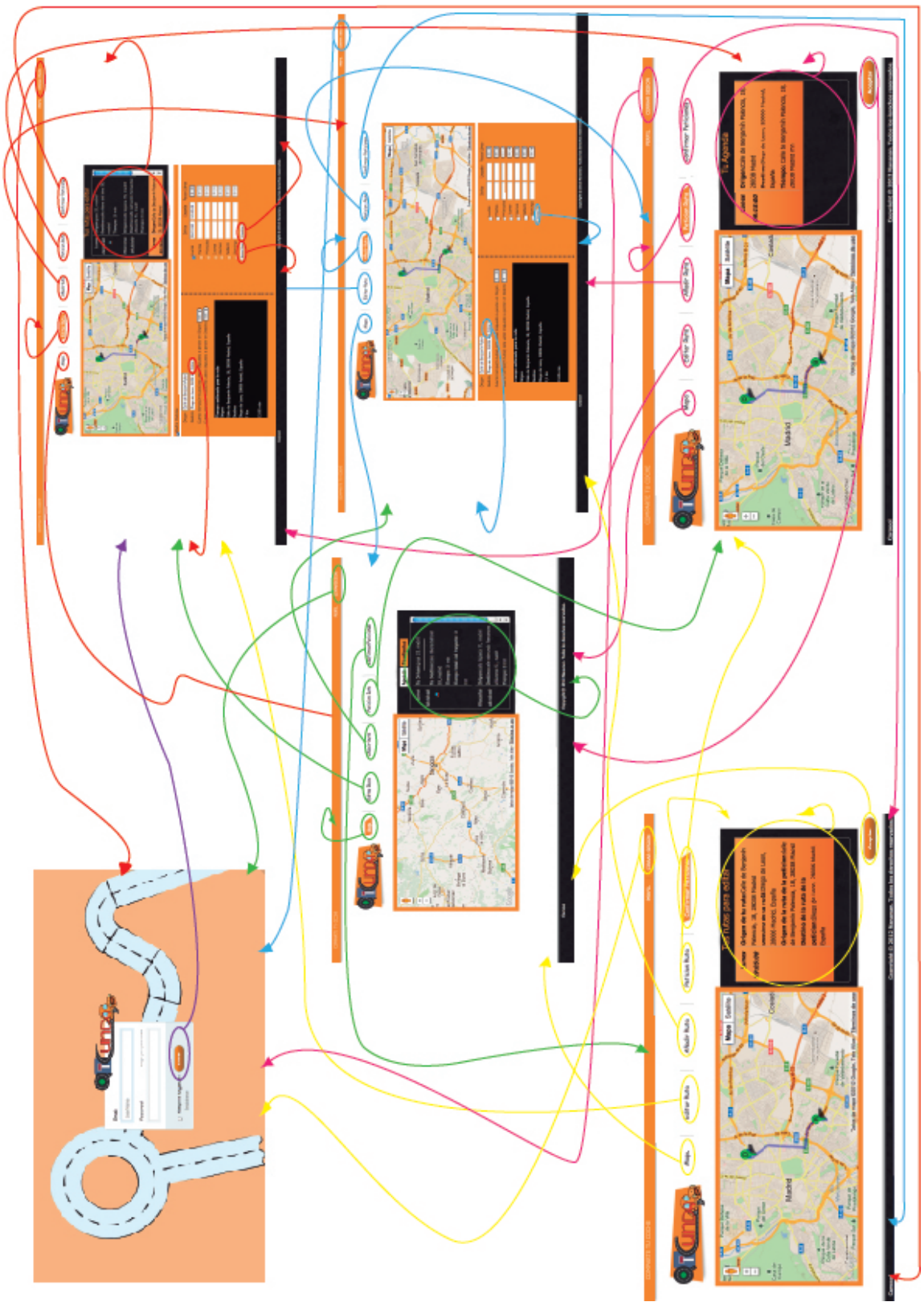
CONFIRMAR RUTA



Pinche en el Cuadro para ver como quedaría la ruta si se acepta la petición. Haga click en el Botón Aceptar para confirmar la petición.

Apéndice B

Mapa Web



Bibliografía

[1] Beck, Kent, Extreme Programming Explained: Embrace Change, Addison-Wesley Professional, 1999

[2] De Castro Korgi, Rodrigo. El universo LaTeX, 2da edición, Universidad Nacional de Colombia, Facultad de Ciencias. Departamento de Matemáticas, Bogotá, 2003.

[3] Proyecto LaTeX http://www.latex-project.org/

[4] Google Developers: Google Maps https://developers.google.com/maps/?hl=es

[5] AJAX: http://es.wikipedia.org/wiki/AJAX

[6] Introducción a AJAX http://www.librosweb.es/ajax/index.html

[7] Versión 3 de Google Maps JavaScript API https://developers.google.com/maps/documentation/javascript/?hl=es

[8] Manual de PHP: http://www.desarrolloweb.com/php/

[9] Página de PHP: http://www.php.net/

[10] JavaServer Pages: http://es.wikipedia.org/wiki/JavaServer_Pages

[11] CGI: http://es.wikipedia.org/wiki/Interfaz_de_entrada_comun

[12] Artículo sobre Ruby on rails http://sobrerailes.com/pages/en_marcha_con_rails/

[13] ASP.NET: http://es.wikipedia.org/wiki/ASP.NET

[14] Manual de JavaSript http://www.desarrolloweb.com/javascript/

[15] Adobe Flash http://en.wikipedia.org/wiki/Adobe_Flash

[16] Microsoft Silverlight http://es.wikipedia.org/wiki/Microsoft_Silverlight

[17] MySQL http://www.mysql.com/

[18] Oracle Database http://es.wikipedia.org/wiki/Oracle_Database

[19] The Apache Software Foundation http://www.apache.org/

[20] Exim Internet Mailer http://www.exim.org/

[21] The Postfix Home Page http://www.postfix.org/

[22] Exim Internet Mailer http://www.exim.org/

[23] Manual SVN por línea de comandos http://www.hasheado.com/usando-subversion-desde-la-linea-de-comandos.html

[24] Adobe Dreamweaver http://www.adobe.com/es/products/dreamweaver.html

[25] XAMPP http://www.apachefriends.org/es/xampp.html

[26] MySQL Workbench http://www.mysql.com/products/workbench/

[27] TeXShop http://pages.uoregon.edu/koch/texshop/

[28] Patrón Modelo-Vista-Controlador http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

[29] Manual MVC con PHP http://tednologia.com/mvc-en-php/

[30] Patrón Front-Crontroller http://petra.euitio.uniovi.es/~i6950404/wiki/pmwiki.php?n=Tema5.PatronFront-Controller

[31] Fórmula de Haversine http://es.wikipedia.org/wiki/Formula_del_Haversine

[32] La arquitectura MVC http://www.librosweb.es/jobeet_1_3/capitulo4/la_arquitectura_mvc.html

[33] MVC para dispositivos móvilles http://modeladomvc.wikispaces.com/Utilizar+MVC+para+dispositivos+moviles